ENTORNO WEB PARA LA COMPARACIÓN DE METAHEURÍSTICAS INSPIRADAS EN INTELIGENCIA DE ENJAMBRE

BEATRIZ GÓMEZ CARRERO
DANIEL CORRALES RODRÍGUEZ
FEDERICO GARCÍA ÁVILA

PROYECTO DE SISTEMAS INFORMÁTICOS
FACULTAD DE INFORMÁTICA
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN
UNIVERSIDAD COMPLUTENSE DE MADRID

INGENIERÍA INFORMÁTICA

CURSO 2013-2014

Director: Fernando Rubio Diez

AUTORIZACIÓN DE DIFUSIÓN Y UTILIZACIÓN

Se autoriza a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Firmado:

Beatriz Gómez Carrero

Daniel Corrales Rodríguez

Federico García Ávila

ÍNDICE

ÍNDICE DE FIGURAS

RESUMEN

Este proyecto está constituido por una página web que permite la ejecución de distintas metaheurísticas inspiradas en la inteligencia de enjambre con el objetivo de optimizar distintos tipos de funciones, tanto sobre dominios continuos como discretos. Estas distintas ejecuciones pueden ser modificadas mediante múltiples parámetros y permiten la visualización de los distintos resultados obtenidos, de forma tanto gráfica como estadística. Dentro de las ejecuciones, se permite la obtención de distintas comparativas, que permiten al usuario, no solo estudiar los distintos métodos por separado, si no que ofrece la posibilidad de comparar los resultados de varias ejecuciones entre ellas.

Se ha conseguido, por tanto, una aplicación web que permite el estudio de numerosos métodos de optimización, en distintas funciones de minimización, con la posibilidad de ver gráficamente el comportamiento de las poblaciones de cada uno de estos métodos, que permite a usuarios con un conocimiento de dichos métodos tanto amplio, como más reducido, entender el funcionamiento de estos métodos tras variar los distintos parámetros que definen tal comportamiento.

PALABRAS CLAVE

Entorno web, metaheurísticas, computación inspirada en la naturaleza, inteligencia de enjambre, funciones de minimización, optimización de funciones.

ABSTRACT

This project is formed by a web page that allows the execution of different metaheuristics inspired by swarm intelligence with the goal to optimize different kinds of functions, both of continuous and discrete domains. These various executions can be modified by a multiple number of parameters and allow the visualization of the different results obtained, both in a graphic and statistic way. Within the executions, the obtaining of different comparatives is allowed, which enable the user not only to study the different methods separately but also offer the possibility of comparing the results of different executions and methods altogether.

In conclusion, a web application that allows the study of different optimization methods has been accomplished. In addition, it enables to execute these algorithms with different minimization functions, also being able to see the behavior of each of these methods populations graphically, which will allow users with both wide or narrow knowledge of these methods understand how they work after modifying the various parameters that define such behavior.

KEYWORDS

Web environment, metaheuristics, nature-inspired computation, swarm intelligence, minimization functions, function optimization.

1. INTRODUCCIÓN

El problema que se aborda en este proyecto es el de la resolución de problemas metaheurísticos mediante diferentes algoritmos de optimización. No obstante, resolutores de este estilo ya estaban programados con anterioridad aunque siendo su utilización poco usable e intuitiva.

La idea principal es la de acercar la funcionalidad de dichos algoritmos al usuario medio y en particular al estudiante. Para lograrlo se ha hecho uso de la herramienta más accesible y usada hoy en día por todo el mundo, las páginas web.

Creando una interfaz de usuario sencilla e intuitiva a través de una aplicación web, el usuario solo necesita introducir una serie de parámetros y un par de *clics* para ejecutar cualquiera de los algoritmos de optimización. Para ayudar además a la comprensión de los resultados, se realiza una representación gráfica de cómo se comporta cada individuo en su algoritmo correspondiente por lo que resulta una herramienta útil para aquellas personas que se encuentren estudiando o tratando de comprender el funcionamiento de dichos algoritmos, así como para aquellas personas que se dedican a impartir la docencia de los mismos.

¿Pero por qué teniendo reunidos sobre una misma aplicación tantos algoritmos de optimización se ha de quedar la funcionalidad en la mera ejecución de los mismos? Una utilidad extra que se intuye rápidamente es la de realizar estudios estadísticos sobre dichos algoritmos. Una serie de cálculos sobre una consecución de ejecuciones del mismo algoritmo, o de varios algoritmos, proporciona una amplia información interesante acerca de la eficiencia de los mismos para determinados problemas en cuanto a calidad de resultados, tiempo de ejecución, etc. Dicha información mostrada de nuevo gráficamente permite al docente facilitar el proceso de aprendizaje y entendimiento del funcionamiento de los algoritmos para el estudiante.

Por último, se permite al usuario descargar en su máquina los resultados de dichas ejecuciones estadísticas, pudiendo ser consultadas o comparadas con la misma aplicación web en cualquier momento.

1.1. Clasificación de los Problemas

En este proyecto se trabaja con dos tipos de problemas. Por un lado, los problemas que trabajan con funciones sobre dominios continuos, de tipo $f: \mathbb{R}^n \rightarrow \mathbb{R}$, siendo n el número de dimensiones que tiene el dominio de la función, siendo este dominio un conjunto de valores no numerable. El objetivo es encontrar los valores del dominio que al ser sustituidos en una función dada, devuelven el menor (o mayor) valor del rango de dicha función.

Para la resolución de dicho problema se utilizan metaheurísticas, que resuelven este problema mediante la implementación y utilización de ciertos algoritmos, a los que se les modifican ciertos parámetros proporcionados por el usuario, para ayudar a dicho fin. Algunas de estas metaheurísticas son los algoritmos PSO (Particle Swarm Optimization) [1, 14], SSO (Social Spider Optimization) [3] o ABC (Artificial Bee Colony) [2, 4, 15, 16], entre otros.

Por ejemplo, un tipo de función de entrada puede ser una función $f(x) = x^2$, y el objetivo, encontrar el valor de *x* que produce que el resultado de esa función sea mínimo, es decir, $x = 0$. Esta función puede ser mucho más compleja, además de contar con un dominio cuyo número de dimensiones puede ser mucho mayor, donde cada dimensión se mueve dentro de un intervalo dado. Por ejemplo, la función Ackley (véase, por ejemplo [5]) posee numerosos mínimos locales, que pueden dificultar el encontrar los valores de x que devuelvan el mínimo valor de la función global:

$$f(x) = 20 + e - 20 \cdot \exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{30}\sum_{i=1}^{n} \cos 2\pi\, x_i\right), -2 \leq x_i \leq 2$$

cuyo mínimo es 0, cuando cada una de las dimensiones toma el valor 0, es decir, $\min\big(f(x) = (0, \dots, 0)\big) = 0$. Siendo n el número de dimensiones de la función. Una gráfica de este algoritmo, para un número de dimensiones $n = 2$ puede verse en la Figura 1.1:

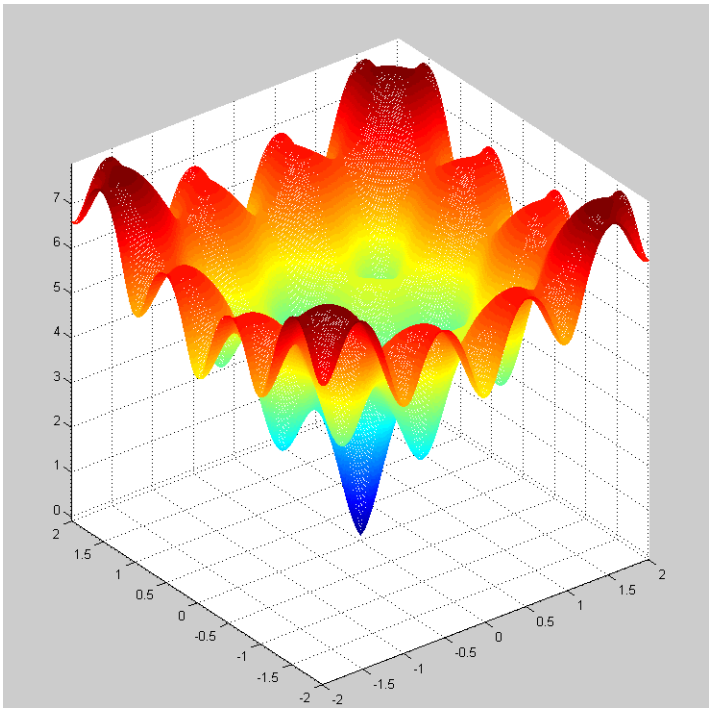

Figura 1.1: Función Ackley de dos dimensiones

Por otro lado, en este proyecto también se trabaja con un tipo de problemas muy diferente, al considerar la optimización de funciones discretas. Esto es, el dominio de la función a optimizar es un conjunto de valores numerable.

Algoritmos que trabajan con este tipo de funciones son, entre otros, ACO (Ant Colony Optimization), que está inspirado en cómo las hormigas encuentran caminos de bajo coste, o RFD (River Formation Dynamics) que se basa en cómo el agua encuentra buenos caminos para descender hasta el mar.

Un problema muy característico que puede resolverse mediante ACO, es el problema del viajante, TSP (Travel Salesman Problem) [9, 17, 18]. Dada una lista de ciudades y las distancias entre ellas, busca la solución que devuelve el camino más corto posible, que visita cada una de las ciudades una vez y vuelve a la ciudad de origen.

2. METAHEURÍSTICAS DE OPTIMIZACIÓN BASADAS EN INTELIGENCIA DE ENJAMBRE

Los algoritmos de inteligencia de enjambre son algoritmos de inteligencia artificial. Este tipo de algoritmos se basa en el comportamiento de ciertos animales, y en su forma de relacionarse con el ambiente y con los individuos que les rodean, ya sea de forma local o global.

Cada individuo de la población representa una posición en el espacio de búsqueda, por lo tanto, tiene un número de dimensiones que se corresponde con el número de dimensiones del dominio de la función a optimizar.

El objetivo que se persigue es la optimización (minimización o maximización) de una función dada. Para ello, mediante una serie de iteraciones, cada individuo de la población modificará su comportamiento dependiendo de distintos parámetros, con el objetivo de minimizar (o maximizar) el valor de la función dada, también llamada función de *fitness*.

Estos algoritmos permiten la modificación de diversos parámetros, que determinan el comportamiento de los individuos, así como el funcionamiento del propio algoritmo. A continuación se describen 4 metaheurísticas, las 3 primeras sobre dominios continuos (PSO, SSO y ABC) y la última sobre dominios discretos (ACO).

2.1. Optimización por enjambre de partículas (PSO)

El funcionamiento de este algoritmo se basa en la búsqueda de una solución óptima mediante el movimiento de los individuos por el espacio de búsqueda, con una cierta velocidad, influenciados por otros individuos de su vecindario.

Un vecindario está formado por distintos individuos de la población, y cada individuo tiene su propio vecindario. Se distinguen distintos tipos de topologías para formar los vecindarios [7]. Entre ellos:

- Global Best PSO: El vecindario está formado por todos los individuos de la población.
- Local Best PSO: El individuo tiene dos vecinos, el creado inmediatamente antes y el creado inmediatamente después de dicho individuo. El primer y el último individuo, por tanto, solo tienen un vecino (el segundo y el penúltimo individuo, respectivamente).

Figura 2.1: Topología del Local Best PSO

- Von Neumann PSO: Los individuos se colocan formando una red cuadrada, ordenada por el momento de creación de cada uno de los individuos. De esta forma, cada uno de los individuos tendrá como vecinos los situados arriba, abajo, a su derecha y a su izquierda con respecto al

momento de creación de los mismos. Por tanto, los individuos situados en la parte más externa de la red, tienen tres o menos vecinos.

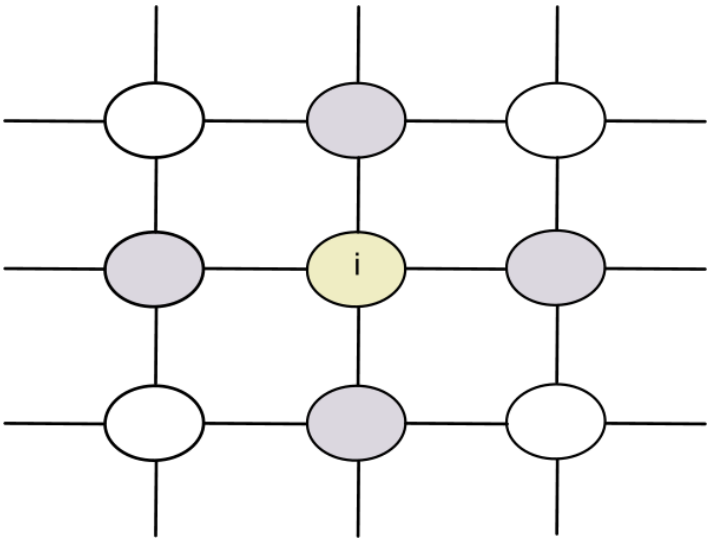

Figura 2.2: Topología del Von Neumann PSO

PSO permite la elección tanto del número de individuos que forman la población, como del número de iteraciones máximas permitidas para la búsqueda de la solución óptima. Además se permite la modificación de dos parámetros $\boldsymbol{\varphi_1}$ *y* $\boldsymbol{\varphi_2}$, que influencian el cambio de dirección y velocidad de cada individuo dando mayor o menor peso a las informaciones locales o globales. Para cada iteración se guarda el mejor individuo de la población actual (*mejor local*), además del mejor individuo encontrado hasta el momento en el vecindario (*mejor global*). Un individuo es mejor que otro si tiene un valor de *fitness* mayor.

Una explicación más precisa del funcionamiento se describe como sigue:

Cada individuo representa una posición en el espacio de búsqueda, y se desplaza por dicho espacio a una determinada velocidad.

Se crea una población inicial, donde el valor de cada dimensión del individuo se mueve aleatoriamente entre un rango mínimo y un rango máximo dados.

Se producen una serie de iteraciones en las que se modifican la velocidad y la posición de cada individuo. Estas iteraciones se realizan hasta llegar a la condición de terminación del algoritmo. Esta condición de terminación puede ser alcanzar el número máximo de iteraciones dadas al inicio de la ejecución del algoritmo, llegar a un número de iteraciones que no produzca una mejora significativa en la mejor solución encontrada, etc.

La velocidad del individuo en la iteración $\boldsymbol{k+1}$ viene determinada por:

La velocidad del individuo en la iteración $\boldsymbol{k}$, con un factor de influencia $\boldsymbol{\omega}$ (peso inercial), que determina la proporción en la que se tiene en cuenta la velocidad actual para el cálculo de la velocidad posterior del individuo.

Una información global determinada por la diferencia entre la posición actual del individuo y la posición del mejor individuo encontrado hasta el momento $\boldsymbol{g}$ con un factor de influencia determinado por $\boldsymbol{R1}$ y $\boldsymbol{\varphi_1}$, siendo $\boldsymbol{R1}$ un valor real comprendido entre 0 y 1 y $\boldsymbol{\varphi_1}$, un parámetro de entrada definido por el usuario.

Una información local determinada por la diferencia entre la posición actual del individuo y la posición de la mejor partícula del vecindario de dicho individuo, con un factor de influencia determinado por $\boldsymbol{R2}$ y $\boldsymbol{\varphi_2}$, siendo $\boldsymbol{R2}$ un valor real comprendido entre 0 y 1 y $\boldsymbol{\varphi_2}$, un parámetro de entrada definido por el usuario. Dicha velocidad se modifica siguiendo la siguiente fórmula:

$$\boldsymbol{v^{k+1} = \omega \cdot v^k + \varphi_1 \cdot R1 \cdot (pBest - x^k) + \varphi_2 \cdot R2 \cdot (g - x^k)}$$

La nueva velocidad está determinada por la suma de las influencias local, global y del propio individuo.

La posición del individuo en la iteración $\boldsymbol{k+1}$, $\boldsymbol{x^{k+1}}$ viene determinada por la posición en la que está situado actualmente, $\boldsymbol{x^k}$ más el desplazamiento que supone el movimiento de una partícula con velocidad $\boldsymbol{v^{k+1}}$ durante una unidad de tiempo. Dicha posición se modifica siguiendo la siguiente fórmula:

$$\boldsymbol{x^{k+1} = x^k + v^{k+1}}$$

Una vez se tiene la nueva posición del individuo en el espacio de búsqueda se calcula el nuevo valor de *fitness* para esa posición, y se actualizan los valores de los mejores individuos locales y globales de la población actual.

2.2. Optimización por arañas sociales (SSO)

En SSO se busca el individuo que representa una solución del espacio de búsqueda cuya función de *fitness* es mínima (o máxima). La población está formada por un número dado de individuos, que representan un grupo de arañas que en función de sus características y distancia que las separan producen unas vibraciones que pueden atraer o repeler a otros individuos en mayor o menor medida.

Una explicación más precisa del algoritmo sigue los siguientes pasos y características:

Cada individuo s_i, representa el comportamiento de una araña, que a su vez tiene la característica de ser una solución a una función de optimización dada. Cada una de estas arañas tiene un género (masculino o femenino). Además, en función del valor de *fitness* de dicho individuo, $J(s_i)$, del valor de *fitness* del mejor individuo, $best_s$, así como del valor de *fitness* del peor individuo de la población, $worst_s$, se calcula el peso que tiene dicha araña, w_i , siguiendo la fórmula

$$w_i = \frac{J(s_i) - worst_s}{best_s - worst_s}$$

siendo 0.0 el valor del peso para la araña cuyo valor de *fitness* es el mayor de toda la población, y 1.0 el valor del peso para el individuo cuyo valor de *fitness* es el menor de todos los individuos de la población (para funciones de minimización).

El número de arañas hembra N_f se mueve entre el 65% y el 90% del número de arañas totales N. Para calcular N_f, se calcula un número aleatorio que se mueve entre dicho rango, siguiendo la fórmula

$$N_f = floor\big((0.9 - rand \cdot 0.25) \cdot N\big)$$

donde $rand$ es un número real aleatorio comprendido en el rango [0.0, 1.0), y $floor$ es una función que trunca a un número entero el resultado del cálculo del número de arañas hembra.

El número de arañas macho N_m, será, por tanto

$$N_m = N - N_f$$

Al inicio del algoritmo, se da valor a cada dimensión de cada individuo del espacio de búsqueda de forma aleatoria, donde dicho valor está comprendido entre los extremos inferior y superior dados para la dimensión.

Dos individuos s_i y s_j están separados entre ellos una distancia $d_{i,j}$ calculada como la distancia euclídea entre las dos posiciones del espacio de búsqueda que representan por cada uno de los individuos, siguiendo la fórmula

$$d_{i,j} = \| s_i - s_j \|$$

Cada individuo i recibe vibraciones de otro individuo j de la población, siguiendo la fórmula

$$Vib_{i,j} = w_j \cdot e^{-d_{i,j}^2}$$

donde w_j es el peso del individuo j (comprendido entre 0.0 y 1.0). La parte de la fórmula $e^{-d_{i,j}^2}$ garantiza que cuanto mayor sea la distancia entre los individuos i y j menor será la influencia que ejerce el individuo j sobre el individuo i. Además, cuanto mejor sea el valor de *fitness* del individuo j, mayor será el valor de la vibración que se propague al individuo i.

Se tienen en cuenta las vibraciones que un individuo i percibe de otro individuo j en tres casos:

- Cuando j es un individuo caracterizado por la cualidad de ser el individuo más cercano a i cuyo peso es mayor que el de i, es decir $w_j > w_i$. En este caso la vibración captada por el individuo i se llama $Vibc_i$ y el individuo j se llama individuo c.

- Cuando j es un individuo caracterizado por la cualidad de ser el mejor individuo de la población, esto es, que posee el mayor peso de todos los individuos, es decir $w_i = 1.0$. En este caso la vibración captada por el individuo i se llama $Vibb_i$ y el individuo j se llama individuo b.

- Cuando j es un individuo caracterizado por la cualidad de ser el individuo de género femenino más cercano al individuo i. En este caso la vibración captada por el individuo i se llama $Vibf_i$ y el individuo j se llama individuo f.

El movimiento de las hembras está determinado por un parámetro elegido por el usuario, PF que determina la probabilidad de atracción o repulsión del individuo actual f_i en la iteración k, f_i^k respecto a los individuos b y c. Para ello, se genera un valor aleatorio r_m comprendido en el intervalo [0.0, 1.0] que determina dicho movimiento, siguiendo la fórmula

$$f_i^{k+1} = \begin{cases} f_i^k + \alpha \cdot Vibc_i \cdot \left(c - f_i^k\right) + \beta \cdot Vibb_i \cdot \left(b - f_i^k\right) + \delta \cdot \left(rand - \frac{1}{2}\right), & si\ r_m < PF \\ f_i^k - \alpha \cdot Vibc_i \cdot \left(c - f_i^k\right) - \beta \cdot Vibb_i \cdot \left(b - f_i^k\right) + \delta \cdot (rand - \frac{1}{2}), & si\ r_m \geq PF \end{cases}$$

Este movimiento tiene una gran componente aleatoria, introducida por los valores α, β, δ y $rand$, que son valores reales aleatorios comprendidos entre [0.0, 1.0].

Por otro lado, los individuos machos de la población se dividen en dos clases: los machos dominantes y los machos no dominantes. Para determinar a qué grupo pertenece cada uno de los individuos machos de la población, se ordenan los individuos en función de su peso w. Se detecta el individuo situado en la mediana, es decir, aquel individuo situado en la posición que deja igual número de individuos a su derecha y a su izquierda. Los individuos que posean un peso mayor que el del individuo situado en la mediana se consideran machos dominantes, esto es, los individuos dominantes son aquellos que tienen un mejor *fitness* que el resto. Por otro lado, los individuos restantes componen el grupo de machos no dominantes.

Los machos dominantes modifican su movimiento en función de la posición de la hembra más cercana a él, o individuo f. Este movimiento es un movimiento de atracción, que introduce una componente aleatoria determinada por los parámetros α, δ y $rand$, que son valores reales comprendidos entre 0 y 1. Dicho movimiento sigue la siguiente fórmula:

$$m_i^{k+1} = m_i^k + \alpha \cdot Vibf_i \cdot \left(f - m_i^k\right) + \delta \cdot \left(rand - \frac{1}{2}\right)$$

Donde m_i^k es el macho situado en la posición i respecto al valor de peso del resto de individuos macho en la iteración k.

Los individuos no dominantes, en cambio, modifican su posición teniendo en cuenta al resto de individuos macho, moviéndose poco a poco a una posición situada entre todos ellos. Una componente aleatoria se introduce con el valor α comprendido entre 0 y 1.

$$m_i^{k+1} = m_i^k - \alpha \cdot \left(\frac{\sum_{h=1}^{N_m} m_h^k \cdot w_{N_h}}{\sum_{h=1}^{N_m} w_{N_h}} - m_i^k\right)$$

Donde w_{N_h}es el peso de individuo situado en la posición h por orden de peso, y m_i^k es el macho situado en la posición i por orden de pesos en la iteración k.

Tras el movimiento de cada uno de los individuos de la población se produce la operación de apareamiento, que busca la creación de un nuevo individuo. Para ello, cada macho dominante m_g crea un grupo de hembras situadas dentro de un cierto rango r o rango de apareamiento. Este rango depende tanto del número de dimensiones de la función a optimizar n, como de los valores de los rangos mínimo y máximo de cada dimensión j, p_j^{low} y p_j^{high} respectivamente. Este rango r se calcula siguiendo la fórmula

$$r = \frac{\sum_{j=1}^{n}(p_j^{high} - p_j^{low})}{2 \cdot n}$$

Las arañas hembra situadas a una distancia del macho dominante actual menor que r se añaden al grupo E^g. Si el grupo E^gno tiene ninguna hembra, el proceso de apareamiento se cancela. En caso contrario, nace una nueva cría cuya posición en el espacio de búsqueda depende tanto de E^g como del macho dominante m_g, cuya unión forma el grupo T^g. Cada individuo del grupo T^g tiene una probabilidad de influencia mayor cuanto mayor es su peso, y menor cuanto menor es su peso proporcional a su peso y dicha probabilidad está determinada por el método de la ruleta, que se describe como sigue. A cada una de las dimensiones del espacio de búsqueda que representa el nuevo individuo, se le da el valor (de esa misma dimensión) que posee uno de los componentes del grupo T^g. Para saber cuál de los individuos del grupo T^g es el elegido, se calcula la probabilidad de ser escogido Ps_i, siguiendo la fórmula $Ps_i = \frac{w_i}{\sum_{j \in T^g} w_j}$, es decir, la probabilidad de ser escogido es el peso del individuo dividido entre la suma de los pesos de todos los individuos del grupo T^g.

Una vez se da valor a cada una de las dimensiones de la nueva cría, se calcula el valor de *fitness* que posee. En caso de que el valor de *fitness* del nuevo individuo sea mejor que el *fitness* del peor individuo de la población actual, la nueva cría sustituye al peor individuo en posición y género; en caso contrario, la población no se altera y la nueva cría se descarta.

2.3. Optimización por colonias de abejas (ABC)

El algoritmo ABC tiene como objetivo encontrar la solución del espacio de búsqueda que devuelve un valor de función de *fitness* óptimo mediante minimización (o maximización). Para ello simula el comportamiento de un enjambre de abejas, que buscan las mejores fuentes de comida en el espacio de búsqueda proporcionado. Para lograrlo, ha de seguir un cierto comportamiento.

Cada una de las soluciones del espacio de búsqueda de la solución a optimizar se corresponde con una posible fuente de comida en ABC, y el valor de *fitness* de dicha solución es la cantidad de néctar o alimento que posee dicha fuente de comida.

La población está formada por una colonia de abejas de un tamaño dado como parámetro. Los individuos de dicha población se dividen en dos grupos principales: las abejas empleadas y las abejas no empleadas.

Las abejas empleadas tienen como cometido la búsqueda de nuevas fuentes de alimento cercanas a las fuentes de alimento que están explotando actualmente. Si un cierto individuo tiene asignada la

fuente de alimento $food_i$, cuyo número de dimensiones viene dado por el número de dimensiones de la función a optimizar, se moverá alrededor de la posición de dicha fuente de alimento para buscar una nueva y calcular cómo de buena es. Para ello, se pueden utilizar diversos métodos de mutación que modifiquen ligeramente dicha posición en el espacio de búsqueda, por ejemplo: se calcula una posición aleatoria

$$r \in [1, n],$$

siendo n el número de dimensiones de la función a optimizar. Se calcula un nuevo valor aleatorio v, comprendido entre los extremos mínimo y máximo de la función a optimizar en la dimensión r. Se asigna el valor v a la posición r de la fuente de alimento actual. Tras esto se calcula cómo de buena o mala es la nueva fuente de alimento mediante el cálculo de su nuevo valor de *fitness*. Si la nueva fuente de alimento es peor que la anterior, se aumenta un contador que determina el número de veces que se ha repetido este paso sin conseguir algún tipo de mejora en la función de *fitness* $trial;$ en caso contrario, la nueva fuente de alimento sustituye a la anterior, y el contador $trial$ se reinicia a 0.

Por otro lado, las abejas no empleadas se dividen a su vez en dos subgrupos: las abejas espectadoras (*onlooker bees*) y las abejas exploradoras (*scout bees*).

Las abejas espectadoras modifican su posición de búsqueda de nuevas fuentes de alimento en función del resto de abejas del enjambre. Cada una de las demás abejas a_i tiene una cierta probabilidad p de influir en la posición de la abeja espectadora, y dicha probabilidad puede ser calculada por diversos métodos en los que esta probabilidad de influencia sea proporcional a lo buena o mala que sea su fuente de alimento actual, así a mayor valor de la función de *fitness*, mayor probabilidad de influir en la posición a buscar de la abeja espectadora. Un posible método para calcular dicha probabilidad, es el método de la ruleta, en el que la probabilidad de ser escogida cada abeja viene dada por la fórmula $Ps_i = \frac{f_i}{\sum_{j \in Pob} f_j}$, donde f_i es el valor de la función de *fitness* de la abeja en la posición i por orden de creación, y j toma el valor de la función de *fitness* de cada una de las fuentes de alimento guardadas actualmente por la población de abejas. Una vez determinada la probabilidad de selección de cada abeja, la nueva posición del espacio de búsqueda se puede determinar de varias formas, una de ellas es asignando el valor de la posición de la fuente de alimento de la abeja seleccionada a_i en la dimensión $d \in [1, n]$ (donde n es el número de dimensiones de la función a optimizar) a la dimensión d de la nueva fuente de alimento. Una vez rellenadas cada una de las dimensiones de la nueva posición de la fuente de alimento, se calcula cómo de buena es dicha fuente de alimento, calculando el valor de la función de *fitness* de dicha posición. En caso de que el valor de *fitness* de la nueva fuente de alimento sea mejor que la función de *fitness* anterior, se sustituye la antigua fuente de alimento por la nueva fuente de alimento y se reinicia el contador $trial$ que indica el número de veces que se ha repetido el paso sin mejorar el valor de *fitness* para la fuente de alimento; en caso contrario se aumenta en uno dicho valor, indicando que se ha repetido una vez más el paso actual sin conseguir mejorar el valor de *fitness* para la fuente de alimento de la abeja.

Por último, las abejas exploradoras o abejas *scout* son aquellas que buscan aleatoriamente por todo el espacio de búsqueda, intentando encontrar nuevas fuentes de alimento que tengan una gran cantidad de recursos que explotar. Para ello, se asigna un valor aleatorio comprendido entre el

rango mínimo y máximo de la función a optimizar para cada una de las dimensiones de dicha función. Se calcula el valor de *fitness* o de bondad de la nueva fuente de alimento y se reinicia el contador de intentos de mejora fallidos $trial$ de la fuente de alimento de dicha abeja.

Cuando el parámetro $trial$ en algún momento alcanza el valor máximo de intentos $limit$ introducido como parámetro en el algoritmo ABC, cualquier abeja, ya sea empleada o no empleada, pasa a ser una abeja exploradora, que intenta encontrar una nueva fuente de alimento de forma aleatoria con el fin de no quedarse estancada en una fuente de alimento que no produce mejoras con el paso del tiempo.

2.4. Optimización por colonia de hormigas (ACO)

El comportamiento que ACO simula, es el comportamiento de las colonias de hormigas a la hora de buscar fuentes de comida con un coste de distancia mínimo en el espacio. Cuando las hormigas se mueven dejan a su paso un rastro de feromonas que permiten al resto de hormigas guiarse para saber qué camino escoger. Con el paso del tiempo, las feromonas que se quedan en el ambiente se evaporan, por lo tanto, cuanto más corto sea el camino a recorrer, el rastro de feromonas a seguir será mucho más fuerte ya que el camino se recorre en un menor tiempo. Por el contrario, si el camino a recorrer es muy largo, el rastro de feromonas se va eliminando, lo que provoca que las hormigas no recorran ese camino, si no que busquen otro con mayor número de feromonas y por tanto más corto.

En ACO el objetivo que se persigue es encontrar los caminos óptimos en un grafo de aristas valoradas. A modo de ejemplo, consideremos cómo aplicar ACO a un problema concreto y bien conocido, como el problema del viajante o TSP (Travelling Salesman Problem) [35], que busca el camino cerrado que recorre cada uno de los nodos de dicho grafo y que tiene un coste mínimo. Un grafo de aristas valoradas es aquel que posee caminos entre los nodos con un cierto coste.

Para la búsqueda de dicho camino, se trabaja con nodos numerados de 1 a n, siendo n el número de nodos totales a visitar. Cada uno de los nodos puede tener una posición concreta que permita calcular el coste de cada camino entre los nodos, o simplemente pueden darse las distancias si no es relevante dónde está situado cada uno de dichos nodos.

Un comportamiento más preciso del algoritmo de la colonia de hormigas aplicado al problema del viajante se describe como sigue.

Se calcula la distancia entre cada uno de los nodos del grafo es decir, el coste de los caminos que unen cada uno de los puntos a visitar. Estos caminos pueden tener un número de feromonas inicial determinado por un parámetro de entrada, o no tener ninguna cantidad de feromonas inicialmente.

A la hora de escoger el camino a seguir para cada hormiga, se calcula cuál es el siguiente nodo a visitar $destino$ dado el nodo en el que está situada actualmente $origen$. Para determinar dicho nodo $destino$ se pueden utilizar múltiples métodos, basándose en muchos posibles parámetros, por ejemplo:

Se puede calcular el siguiente nodo a visitar eligiendo aleatoriamente un nodo de los que no han sido visitados hasta el momento. Al ser una componente tan aleatoria no produciría muy buenos resultados y mucho menos cuando el grafo esté formado por un número muy grande de nodos.

Puede guardarse el mejor camino encontrado hasta el momento, aquel cuyo coste es mínimo. El nodo $destino$ es el siguiente nodo en el camino óptimo que se visita después del nodo $origen$.

Puede escogerse el nodo $destino$ mirando la cantidad de feromonas de los caminos que parten de este nodo $destino$ y que no han sido visitados hasta ahora, y escoger el nodo cuyo camino entre el nodo $origen$ y el nodo $destino$ tiene el número de feromonas mayor.

El método más equilibrado y más usado combina los métodos anteriores, facilitando la búsqueda del mejor camino posible.

Dicho método da una probabilidad p_{od} a cada nodo para ser escogido como el siguiente nodo a visitar calculándose dicha probabilidad con la siguiente fórmula

$$p_{od} = \frac{\left(trail_{od}^{\alpha} \cdot \eta_{od}^{\beta}\right)}{\sum_{d\ \epsilon\ nodos\ a\ visitar}\left(trail_{od}^{\alpha} \cdot \eta_{od}^{\beta}\right)}$$

Donde $trail_{od}$ es la cantidad de feromonas depositadas en el camino que une los nodos $origen$ y $destino$, y η_{od} es la atracción que produce el nodo $destino$ a la hora de ser escogido como siguiente nodo a visitar.

El valor más común que se asocia a η_{od} suele venir asociado a la distancia que hay entre los dos nodos $origen$ y $destino$ de la forma $1/d_{od}$ es decir, a mayor distancia entre los nodos a conectar, menor deseo de recorrer dicho camino.

Por otro lado, el parámetro α determina la influencia que ejerce el rastro de feromonas que hay en el camino entre los nodos $origen$ y $destino$, así como el parámetro β determina la influencia de η_{od} o atracción por el nodo $destino$. Cabe destacar que $\propto \geq 0$, es decir, la cantidad de feromonas puede o no tenerse en cuenta, y que $\beta \geq 1$, es decir, el parámetro de atracción es algo a tener siempre en cuenta.

Una vez las hormigas han decidido cuál es el camino que van a recorrer, dejan un cierto rastro de feromonas a su paso que puede ser calculado de diversas formas (dejando una cantidad constante de feromonas por el lugar donde hayan pasado, teniendo en cuenta la distancia que han recorrido, etc.).

A su vez, cada uno de los caminos del grafo que anteriormente poseían un cierto número de feromonas pierde parte de ese rastro, ya que las feromonas se van evaporando con el tiempo. Esa pérdida de feromonas está determinada por un factor llamado $evaporación$, que indica el ratio de feromonas que se eliminan del ambiente con el tiempo. Así, en cada paso, el número de feromonas que quedan en cada uno de los caminos se determina mediante la fórmula

$$trail_{od} = (1 - evaporación) \cdot trail_{od} + \sum_k \Delta trail_{od}^{k}$$

Donde $(1 - evaporación) \cdot trail_{od}$ es el número de feromonas que quedan en el ambiente del camino entre $origen$ y $destino$ tras la evaporación de dichas feromonas, y $\Delta trail_{od}^{k}$ es la cantidad de feromonas que se añaden al camino entre el nodo $origen$ y $destino$ cuando la hormiga k pasa por dicho camino.

3. TECNOLOGÍAS UTILIZADAS

A continuación se presentan las tecnologías que han determinado en gran medida el desarrollo y avance del proyecto. Se realiza una pequeña descripción de la tecnología, así como una descripción del uso de dicha tecnología en este proyecto.

Java

Java es un lenguaje de programación de alto nivel, concurrente y orientado a objetos. Trabaja con jerarquías de clases. El lenguaje de programación Java no necesita de una máquina de compilación independiente, pues utiliza el propio lenguaje para compilar su código.

Java fue creado por James Gosling y publicado en 1995 [22]. Relacionado con el modelo de programación de C++, pero con un recolector de basura que permite el control automático de asignación y liberación de memoria dinámica.

En el proyecto, el lenguaje Java es utilizado para la implementación del código referente a los algoritmos de optimización, así como de las funciones a optimizar en cada caso y sus correspondientes cálculos estadísticos. Es un lenguaje ampliamente utilizado y aprendido por todos aquellos que deciden iniciarse en el mundo de la programación debido a su sencillez y comodidad a la hora de programar y por estas razones se ha optado por éste para programar la lógica interna del servidor.

Otra ventaja que ofrece el código Java es la de poder ser incrustado en secciones de código HTML utilizando la tecnología JSP, haciendo prácticamente trivial la comunicación cliente-servidor a la hora de solicitar cierta necesidad de computación al servidor.

JavaServer Pages (JSP)

JSP nos permite la opción de crear aplicaciones web compuestas con código HTML. Dichas aplicaciones poseen a su vez una serie de etiquetas que permiten la inclusión de código Java del lado del servidor [23].

El motor de estas JavaServer Pages está basado en los servlets [24], los cuales se pueden definir como *applets* del lado del servidor que responden a peticiones del cliente [25], muy útiles para extender la funcionalidad del servidor web.

El uso práctico que tiene este tipo de tecnología en este proyecto es la de ejercer la comunicación cliente-servidor entre las diferentes vistas, programadas en HTML, y la lógica del servidor o modelo implementado en lenguaje Java. Se podría decir que se trata de un controlador que recibe peticiones del usuario, ejecuta cierta lógica perteneciente al modelo y devuelve el resultado para que este sea mostrado de nuevo al usuario.

HTML

HyperText Markup Language o HTML es un lenguaje de marcado ampliamente utilizado para la creación de páginas web. Es un estándar a cargo de W3C, organización dedicada a la estandarización de la mayoría de tecnologías utilizadas para el desarrollo web.

El primer diseño inicial corre a cargo de Tim Berners-Lee en 1991 describiendo veintidós elementos básicos de los cuales trece aún perduran en HTML 4 [26].

Utilizando una serie de etiquetas, HTML brinda la posibilidad de definir una serie de elementos tales como botones, campos rellenables, desplegables, links, etc. los cuales serán mostrados posteriormente en la página web y permiten al usuario interactuar con la aplicación. No obstante es necesario incluir tecnología CSS para poder aplicar una serie de estilos a esos elementos, la cual será comentada más adelante.

HTML también incluye la opción de incrustar o importar scripts generalmente programados utilizando tecnología JavaScript para la ejecución de código del lado del cliente.

Por todas las características mencionadas anteriormente, HTML es el lenguaje encargado de programar la vista del proyecto. Ofrece una interfaz al usuario para que este se comunique con la aplicación, realizando peticiones al servidor interactuando con sus componentes como los botones, desplegables, etc.

JavaScript

JavaScript o JS es un lenguaje interpretado orientado a objetos, imperativo, débilmente tipado y más conocido como un lenguaje de script para páginas web aunque también se puede incluir en otro tipo de proyectos basados por ejemplo en Node [27, 36].

Su historia comienza a principios de los años 90 cuando las aplicaciones web empezaban a ser cada vez más complejas y una simple tarea como puede ser comprobar que la información de un formulario estaba correctamente introducida se hacía muy lentamente. Ante este tipo de situaciones, el programador Brendan Eich trató de solucionar este problema adaptando otras tecnologías como ScriptEase para que se pudiera ejecutar código en el propio navegador, en aquel entonces era Netscape Navigator 2.0.

Inicialmente este lenguaje fue nombrado como *Livescript* y fue más tarde cuando Netscape firmó un acuerdo con Sun Microsystems para el desarrollo del nuevo lenguaje de programación el cual pasaría a llamarse JavaScript por un sencillo tema de marketing, Java era la palabra de moda en el mundo de la programación [28].

Gracias a esa capacidad de poder ser ejecutado en el propio navegador es posible en este proyecto realizar llamadas desde el cliente a los diferentes JSP controladores, organizar y mostrar las secciones de la web que sean necesarias en cada momento, generar los gráficos resultado de la ejecución de los algoritmos, etc. Además es destacable el uso que se le da para guardar el estado actual de la sesión del cliente, por ejemplo para que la aplicación sepa si está comparando

estadísticas o si simplemente está ejecutando las mismas. Otro ejemplo claro del uso que se le da para este fin es, por ejemplo, el guardado del idioma actual seleccionado por el usuario.

CSS

Cascading style sheets o CSS es un lenguaje diseñado para la definición de estilos en los documentos electrónicos. Se creó bajo la necesidad de desarrollar un lenguaje estándar para ese fin y se tomaron como base los mejores aspectos de dos propuestas diferentes: CHSS (*Cascading HTML Style Sheets*) y SSP (*Stream-based Style Sheet Proposal*). En 1996, el W3C publicó la primera versión denominada "CSS nivel 1" [29].

Para la definición de estilos del proyecto se utiliza un *framework* de trabajo denominado Bootstrap, el cual se ajusta mediante CSS a los intereses de la web para así crear una serie de páginas usables, sencillas y a la par agradables a la vista del usuario.

3.1. Otras utilidades de interés

Además de las anteriores, otras herramientas que han sido necesarias son las siguientes:

amcharts: Librería utilizada para el tratamiento de gráficas en aplicaciones web. Ofrece una amplia gama de opciones, desde graficas de puntos, de barras, lineales, etc. Se ha utilizado para la representación gráfica de las soluciones de las ejecuciones de los algoritmos y sus datos estadísticos [39].

xml-apis.jar y jdom: Librerías utilizadas para el tratamiento de ficheros con formato XML. Estos ficheros cuentan con una serie de etiquetas que permiten ordenar la información y facilitar su acceso posteriormente para, por ejemplo, procesarla y mostrarla en una página web [30].

json-simple-1.1.jar: Librería utilizada para el procesamiento de textos en formato JSON. Dicho formato es utilizado para la serialización de datos estructurados. Tiene la capacidad de representar tipos como números, booleanos, cadenas de caracteres y *nulls* [31].

jackson-all-1.8.10.jar: Librería utilizada para el tratamiento de objetos JSON. Se ha utilizado su utilidad de conversión sencilla desde objeto Java a objeto JSON y viceversa.

bootstrap-3.2.0: Librería utilizada para la asignación de estilos a los diferentes elementos de los ficheros HTML utilizando HTML, CSS y permitiendo extensiones adicionales mediante JS [32]. Con ella se ha dado el aspecto moderno y agradable a la aplicación web.

jQuery: Librería utilizada para simplificar tareas necesarias para que la aplicación web funcione correctamente, tales como modificar elementos de las páginas, acceder a sus valores, responder a peticiones del cliente sin necesidad de refrescar la página, etc. [33]. Ha resultado una librería muy útil a la hora de modificar las páginas web dinámicamente en el proyecto.

Cytoscape web: Librería utilizada para la representación gráfica de los grafos utilizados en las ejecuciones del algoritmo ACO. Con una serie de adaptaciones permite al usuario en la aplicación web el tratamiento de grafos, pudiendo borrar nodos o aristas, añadir nodos o aristas, mover los elementos, etc.

Subversion: Herramienta de control de versiones utilizada para llevar un control del proyecto y poder realizar el trabajo de desarrollo de forma más sencilla y sincronizada.

CloudForge: Servicio de hosting gratuito para repositorios estilo Github [40], SVN [41], etc. Es el encargado de almacenar los archivos del proyecto en internet para que el equipo de desarrollo trabaje sobre el mismo proyecto y lleve un control de las versiones y de las modificaciones que se hacen sobre el mismo.

TortoiseSVN: Cliente que proporciona una interfaz gráfica intuitiva y sencilla para gestionar repositorios SVN sin necesidad de utilizar comandos por consola. Se asigna el repositorio a una carpeta del ordenador y el programa se encarga de notificar los cambios. Te da opciones sobre esa carpeta de actualizarla o de subir los cambios al repositorio.

Apache Tomcat 7.0: Servidor Web con soporte para Servlets y JSPs. Se ha utilizado para contener el proyecto y poder ser testeado tanto en local, como en el servidor real [42].

Eclipse: Programa utilizado como entorno de desarrollo para el lenguaje de programación Java y como plataforma de integración de herramientas. Con los plugins adecuados se puede convertir en un entorno de desarrollo web que permite iniciar el servidor local con un cómodo menú e integrar el servidor, con la lógica Java y los ficheros HTML, JSP y JavaScript [43].

CSV: Formato con forma de valores separados por comas (*comma-separated values*), sencillo de crear y muy útil a la hora de crear ficheros de tablas compatibles con programas como Microsoft Excel [12]. Dicho formato se utiliza en la aplicación web a la hora de generar el fichero que se descarga con los resultados del comparador de algoritmos.

Expr: Paquete utilizado como base para desarrollar el parser, necesario para la evaluación de funciones personalizadas. Se ha incrementado y adaptado su funcionalidad a las necesidades del proyecto [10].

FireBug y Herramienta de desarrolladores de Google Chrome: Herramientas que permiten el explorado de elementos HTML, así como sus códigos JavaScript asociados y la depuración de los mismos. También ofrecen una cómoda consola para el mostrado de errores [44].

Exploradores Web: Exploradores necesarios para el testeo de la aplicación. Permiten el procesado de los ficheros HTML para ser mostrados al cliente y la comunicación entre el cliente y el servidor.

PhotoShop CS5: Programa de tratado de imágenes utilizado para la creación de las mismas añadidas a la maquetación de la web [45].

NotePad++: Editor de textos con funcionalidad para programadores como por ejemplo detección de palabras reservadas, etc. Se ha utilizado para la creación de código fuente [46].

Microsoft Word: Editor de textos utilizado para la creación de la documentación del proyecto.

Gmail: Sistema de correo electrónico utilizado para la comunicación del equipo de desarrollo con el tutor.

WhatsApp: Aplicación móvil de mensajería instantánea, utilizado para la comunicación interna del equipo de desarrollo. Posee la opción de crear conversaciones grupales que permiten la interacción de varios usuarios simultáneamente desde cualquier lugar, solo necesitando el móvil y conexión a internet.

Skype: Programa de comunicación por voz y videoconferencias, utilizado para la comunicación entre los distintos miembros del equipo de desarrollo del proyecto. También permite la transferencia de archivos entre sus usuarios, funcionalidad muy utilizada por el equipo de desarrollo para la compartición de archivos de interés.

TeamViewer: Programa de visualización y control remoto usado para realizar tareas de desarrollo, en equipo y a distancia. Permite la compartición del estado actual de la pantalla de un equipo en tiempo real [47].

4. ENTORNO WEB DESARROLLADO

A continuación se describe la aplicación, así como las distintas funcionalidades de dicha aplicación. También se describe la estructura del proyecto, junto con la distribución de los paquetes que forman el proyecto y el propósito de cada uno de ellos.

4.1. Descripción de funcionalidades

Inicialmente se encuentra una página de bienvenida con información básica referida al proyecto. Como se puede observar, en la parte superior se encuentra un menú en forma de barra que es la principal herramienta para navegar por las diferentes secciones de la web.

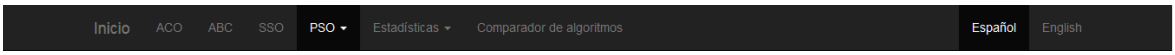

Figura 4.1: Menú principal

Esta barra se mantendrá fija a lo largo de nuestra navegación y como primera funcionalidad básica se encuentra la de seleccionar el idioma. Una vez seleccionado, éste se mantiene a través de las distintas secciones hasta que se decida cambiar explícitamente de nuevo utilizando el menú superior.

Para proseguir con el manual de usuario es necesario agrupar la web en 3 bloques cuya funcionalidad de cara al usuario es prácticamente idéntica:

- Ejecución de problemas
- Ejecuciones estadísticas
- Comparador de algoritmos

Ejecución de problemas

Para acceder a una de las secciones englobadas en este bloque se debe seleccionar una de las opciones siguientes del menú superior.

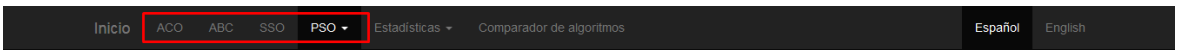

Figura 4.2: Ejecución de problemas

ABC, SSO y PSO

Seleccionando la ejecución de problemas ABC, SSO o cualquiera de los PSO mostrados en el desplegable se abre una página con un bloque dedicado a la inserción de parámetros para el algoritmo, un gráfico y una sección destinada a la explicación breve del funcionamiento de dicho algoritmo.

Figura 4.3: Ejemplo página de ejecución de problemas

Lo primero que se debe seleccionar es el problema que se quiere resolver utilizando el algoritmo seleccionado en el menú superior. Dicho problema se selecciona a través del desplegable que encontramos en el bloque de *Información para el algoritmo*.

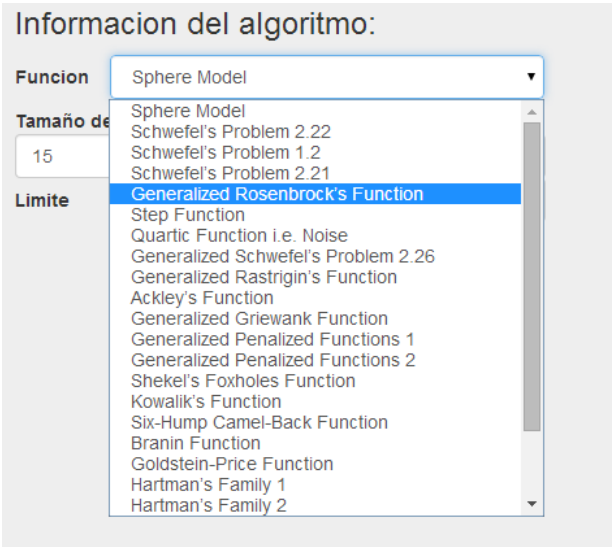

Figura 4.4: Selección de función

Además de poder seleccionar cualquiera de los problemas predeterminados mostrados en el desplegable, se puede introducir manualmente una función para que sea ejecutada seleccionando la opción *Personalizada* del desplegable.

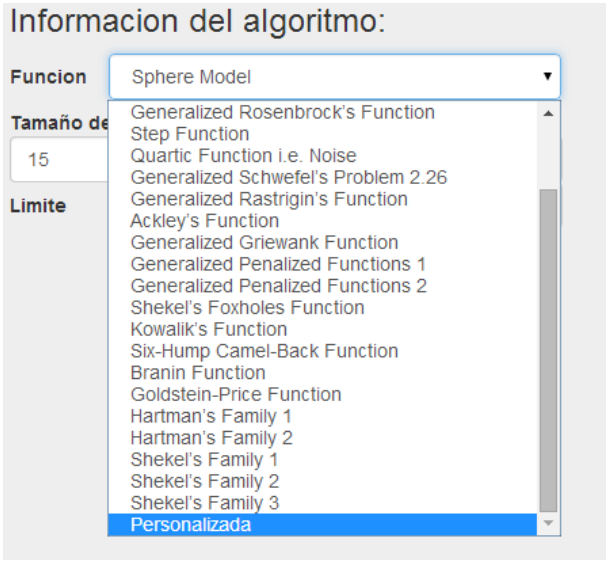

Figura 4.5: Selección de función personalizada

Si se selecciona dicha opción, aparecerá un nuevo bloque que permite al usuario introducir su función personalizada.

Figura 4.6: Introducción de función personalizada

El usuario debe escribir en el campo su función atendiendo a una determinada nomenclatura. Una vez escrita la función es necesario comprobar que la función es válida pulsando el botón *Comprobar!* En caso de no ser correcta se muestra un aviso al usuario indicando los problemas sucedidos y no se permite enviar el problema para su ejecución. De lo contrario, si es correcta, se habilitará el botón de *Enviar*.

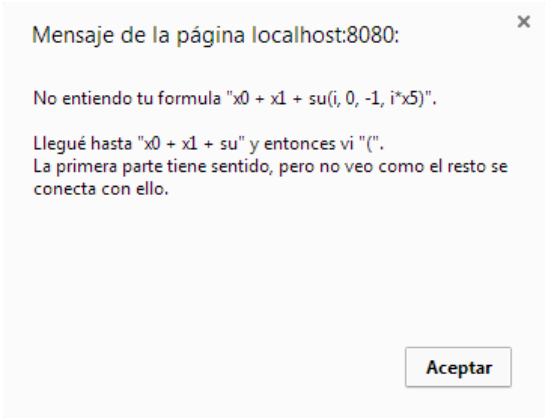

Figura 4.7: Ejemplo error en el parseo

El parser posee una gran variedad de operadores disponibles:

- Operadores aritméticos: +, -, *, /, ^.
- Operadores lógicos: >, <, >=, <=, =, <>.
- Operadores booleanos: and, or
- Operadores trigonométricos: acos(º), asin(º), atan(º), cos(º), sin(º), tan(º)
- Funciones matemáticas: max(nº, nº), min(nº, nº), round (nº), sqrt (nº), ceil (nº), floor (nº), log (nº), sum (variable, índice inferior, índice superior, expresión), prod (variable, índice inferior, índice superior, expresión).

Las variables generales de la función deben seguir el formato xN siendo N un entero positivo y siempre manteniendo orden ascendente. Las variables locales de los sumatorios y productorios (sum y prod) deben seguir el iN pero no es necesario escribirlas en orden.

El resto de campos a completar son los distintos parámetros que requiere cada algoritmo y difieren entre ellos pero todos ellos deben ser campos numéricos. En caso de completar algunos de los campos erróneamente se muestra un mensaje de aviso al usuario.

Una vez completados todos los pasos expuestos anteriormente de forma correcta se envía el problema al servidor para que éste sea ejecutado y se reciba la solución.

Figura 4.8: Botón de enviar

Dicha solución se imprime en una tabla, mostrando las coordenadas del mejor individuo y su *fitness* asociado.

Figura 4.9: Ejemplo de resultado

Si el problema tiene 2 dimensiones se puede observar una representación gráfica de la posición de cada individuo a lo largo de las distintas iteraciones que realiza el algoritmo.

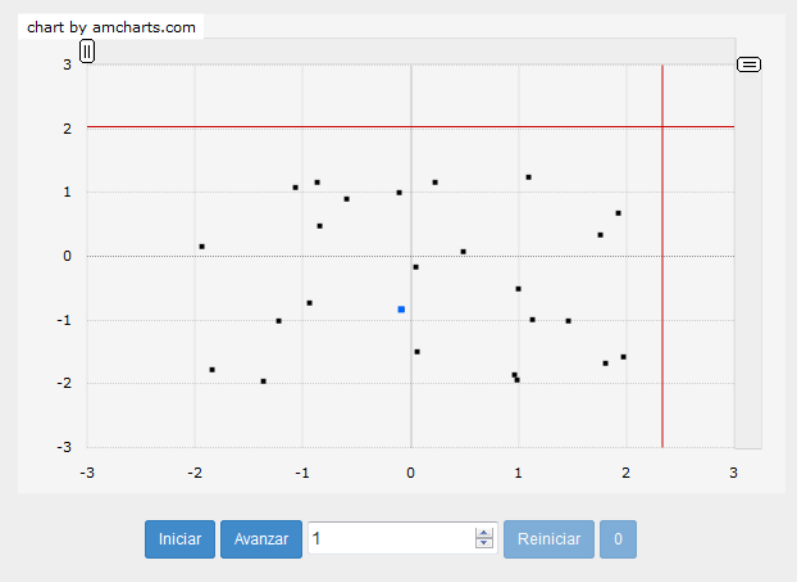

Figura 4.10: Ejemplo de resultado gráfico

Pulsando el botón *Comenzar* comenzará una representación gráfica progresiva a lo largo de las ejecuciones. Para detener dicha ejecución progresiva se deberá pulsar el botón *Parar*.

Figura 4.11: Controles del gráfico

El valor situado a la derecha de este bloque de control del gráfico indica la iteración actual del algoritmo.

Cuando la representación gráfica se encuentra detenida se puede reiniciar pulsando el botón de *Reiniciar*. También permite realizar una representación paso a paso pulsando el botón *Avanzar* y definiendo el número de iteraciones que se desea saltar en cada paso.

El gráfico permite hacer zoom seleccionando el área deseada con el ratón o usando las barras lateral derecha y superior.

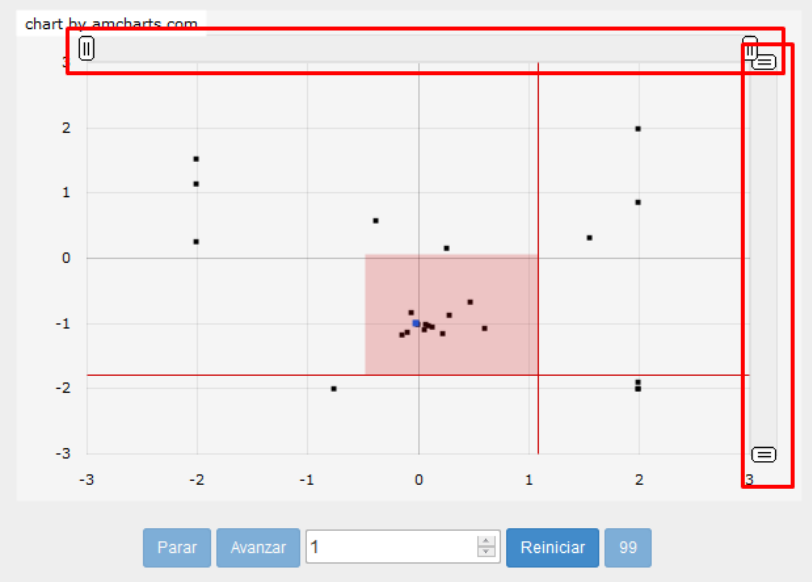

Figura 4.12: Desplazamientos

Para volver a la posición inicial en la que se muestran todos los individuos dentro del gráfico pulsar la tecla *Show all*.

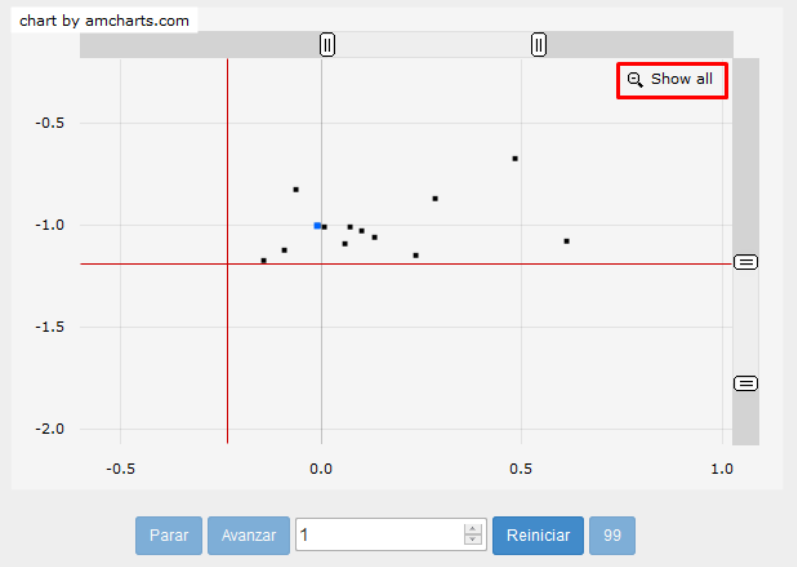

Figura 4.13: Ajustar la vista al gráfico

ACO

Dentro de la ejecución del algoritmo ACO se encuentra un nuevo bloque de funcionalidad: el controlador de grafos.

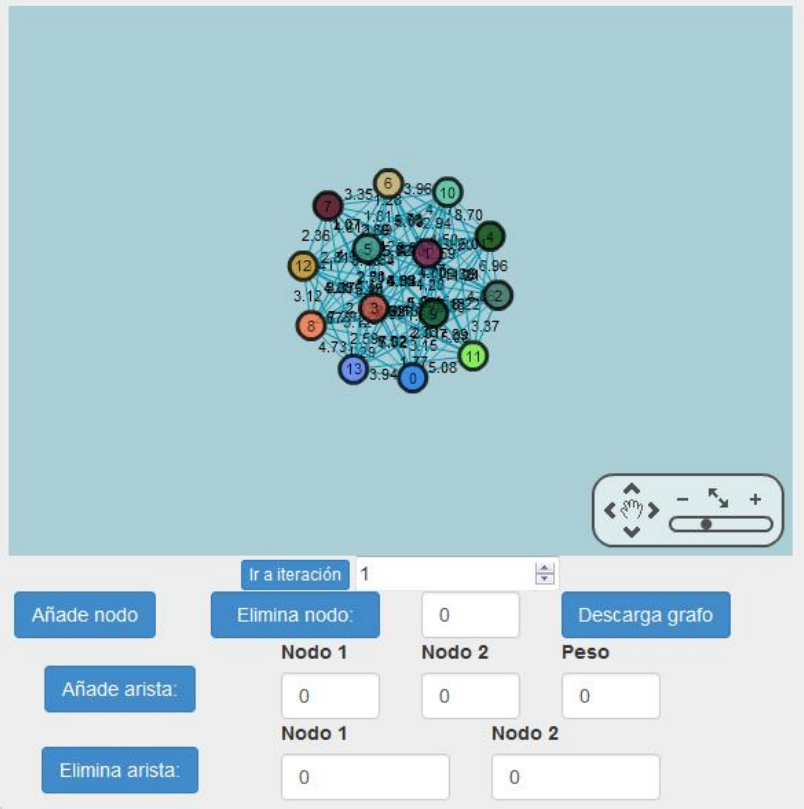

Figura 4.14: Ejemplo de grafo

Aquí se permite la modificación de grafos predeterminados o la creación de uno personalizado en función del grafo seleccionado en el desplegable.

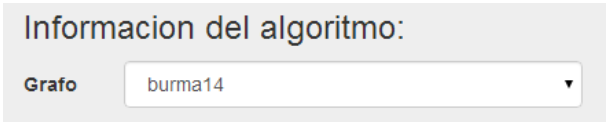

Figura 4.15: Selección de grafo

Como se ve en la imagen anterior se puede añadir nodos (son enumerados automáticamente), borrar un nodo específico, añadir una arista entre 2 nodos existentes especificando su peso y borrar una arista utilizando los nodos situados en sus extremos. Cabe decir que si se borra un nodo que posee aristas, estas también son eliminadas.

Los nodos se pueden desplazar manualmente a lo largo del cuadro azul seleccionándolo y arrastrándolo hasta la posición deseada. Utilizando la barra de opciones situada en la parte inferior derecha del cuadro se puede desplazar a lo largo de dicho cuadro utilizando las flechas o la herramienta de mano. También se permite hacer zoom utilizando los símbolos + y -, desplazando la pequeña barra horizontal o reajustar el grafo al cuadro utilizando la opción *Fit to screen*.


Figura 4.16: Controles de los grafos

Para la correcta ejecución del algoritmo el grafo debe ser conexo, en caso contrario se muestra un mensaje de aviso al usuario.

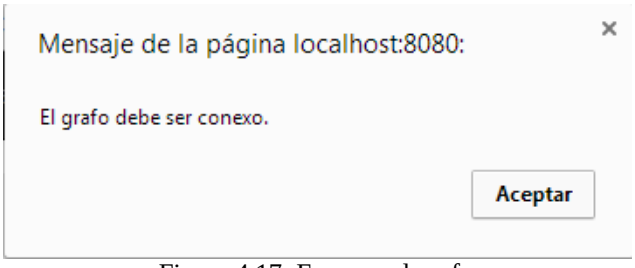

Figura 4.17: Error en el grafo

Una vez completados todos los pasos expuestos anteriormente de forma correcta se envía el problema al servidor para que éste sea ejecutado y se reciba la solución.

Figura 4.18: Botón de enviar

Dicha solución se imprime en una tabla, mostrando el mejor camino encontrado y su coste asociado.

Figura 4.19: Resultados ejecución ACO

Gráficamente se puede seleccionar una iteración para visualizar la solución en dicha iteración. Se muestra en color verde las aristas recorridas por el mejor camino y en azul el resto cambiando la intensidad en función del número de feromonas que se encuentra en cada arista.

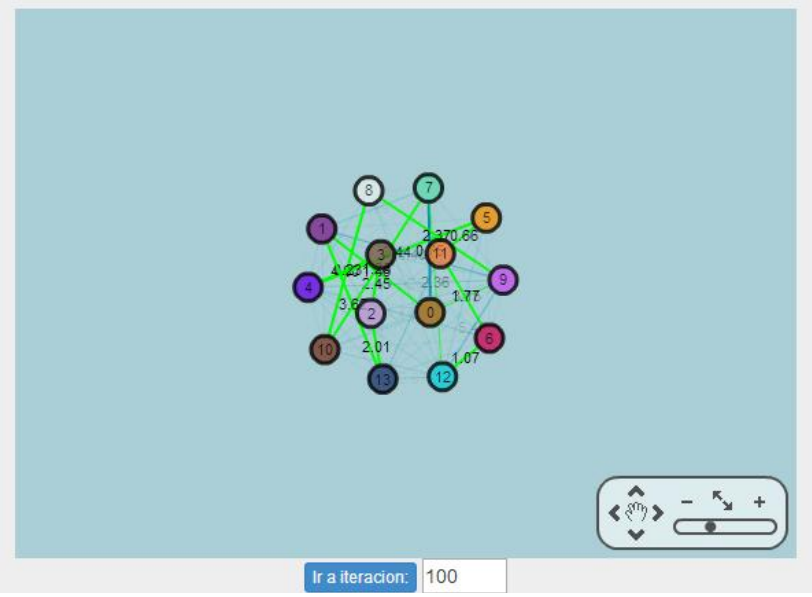

Figura 4.20: Resultados ejecución ACO gráficos

El grafo que se encuentra actualmente en el cuadro se puede descargar en un cómodo fichero pulsando el botón *Descarga grafo*.

Figura 4.21: Botón de descarga de grafo

Posteriormente se puede cargar un grafo seleccionando la opción correspondiente en el desplegable.

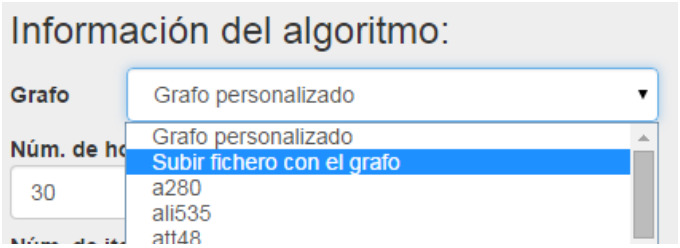

Figura 4.22: Botón de descarga de grafo

Ejecuciones estadísticas

Para acceder a una de las secciones englobadas en este bloque se debe seleccionar una de las opciones siguientes del menú superior.

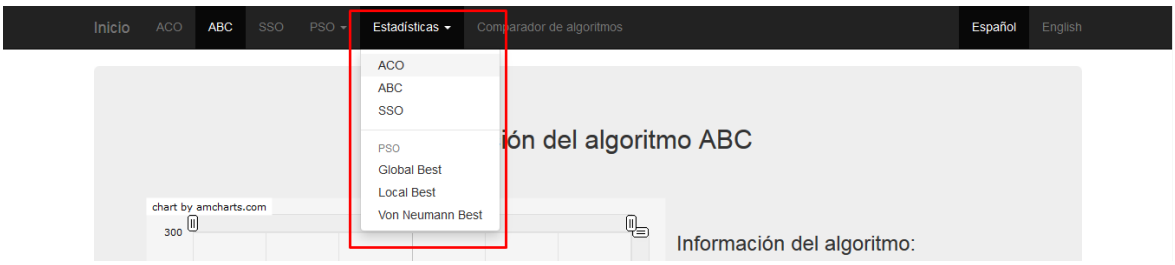

Figura 4.23: Sección de estadísticas

Para los distintos algoritmos se muestran páginas prácticamente idénticas a las encontradas en el bloque de *Ejecución de problemas* aunque la funcionalidad ha cambiado. La idea principal es la de ejecutar el algoritmo un número determinado de veces para calcular valores como el *fitness* medio, la varianza de sus resultados, etc.

Una vez seleccionados los parámetros de la ejecución y el número de ejecuciones deseadas se pulsa el botón *Enviar* para enviar los datos al servidor, realizar las ejecuciones y mostrar el resultado.

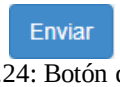

Figura 4.24: Botón de enviar

Dicho resultado se muestra de forma gráfica, incluyendo el *fitness* de cada ejecución y el tiempo consumido para obtener dicho resultado. En una tabla se muestra también la varianza de los distintos *fitness*, el *fitness* medio y el tiempo medio de todas las ejecuciones.

Figura 4.25: Resultado estadístico

Otra nueva funcionalidad que ofrece este bloque es descargar los resultados en formato JSON pulsando el botón *Descargar resultados*. Dicho archivo tiene un nombre descriptivo con el tipo de problema y sus parámetros y se puede utilizar para cargar ejecuciones en el *Comparador de algoritmos*.

Comparador de algoritmos

Para acceder a una de las secciones englobadas en este bloque se debe seleccionar la opción siguiente del menú superior.

Figura 4.26: Comparador estadístico

El objetivo de este bloque es el de comparar ejecuciones entre los distintos tipos de algoritmos PSO, ABC y SSO, o incluso para el mismo algoritmo diferentes combinaciones de parámetros. Con esto se podrá determinar cuál se ajusta mejor para la resolución de un problema, eficiencia, cuál es más rápido ejecutándose, etc.

Lo primero que se debe seleccionar es el problema que vamos a utilizar para las distintas configuraciones de algoritmos seleccionando el deseado, o introduciendo una función personalizada de forma idéntica a los bloques anteriores.

Una vez seleccionado el problema o función se deberán utilizar los botones situados justo debajo para añadir ejecuciones de los algoritmos deseados. Aparecerán entonces diferentes secciones solicitando los parámetros las cuales deberán ser rellenadas para poder realizar la comparativa. En caso de error se pueden borrar pulsando el botón *x* situado en la esquina superior izquierda.

Figura 4.27: Resultado estadístico

El comparador de algoritmos también posee una funcionalidad muy útil: cargar ficheros de ejecuciones guardados por el usuario. Si en alguna de las ejecuciones estadísticas el usuario guardó los resultados puede cargarlos para que se comparen con el resto de ejecuciones pulsando el botón *Elegir archivos* del seleccionador.

Figura 4.28: Cargador de archivos

Una vez seleccionadas y cargadas las diferentes ejecuciones deseadas se debe pulsar el botón *Enviar* para que se realicen los cálculos estadísticos.

Cuando el servidor termina de procesar la información la muestra en una cómoda tabla, incluyendo un nombre descriptivo para que cada ejecución se pueda localizar correctamente. Por último cabe la posibilidad de descargar los resultados de la comparativa en formato CSV.

Figura 4.29: Resultados comparador

4.2. Principales consideraciones técnicas

El principal problema técnico al inicio del proyecto fue el desconocimiento de las tecnologías que debían ser utilizadas. Se hizo una investigación sobre distintas tecnologías, para escoger el lenguaje de programación a utilizar para la implementación de los algoritmos de optimización, así como para realizar la conexión entre la web y la plataforma de implementación escogida. El lenguaje de programación que se decidió para la parte del servidor fue Java y se conectó con la aplicación web mediante JSP.

Una vez implementado el primer algoritmo, surge para el resto de algoritmos a implementar la necesidad de adaptación de todos los algoritmos a un mismo formato de problema a optimizar, así como de la solución a devolver en cada uno de los casos.

Inicialmente se escogió como formato de envío de datos entre servidor y cliente XML. Este formato conllevaba trabajo extra a la hora de generarlo y leerlo posteriormente por lo que más adelante se investigó otro tipo de formato llamado JSON.

Otra gran decisión que se tuvo que tomar fue la de si se debía implementar una librería para el dibujo de las gráficas de las ejecuciones de los algoritmos PSO, SSO y ABC o adaptar alguna librería de representación de gráficas de forma que fuese capaz de dibujar nuestra representación móvil de individuos en el espacio de dos dimensiones. Para ello se decidió adaptar finalmente el uso de la librería amcharts con un control de iteraciones implementado en JavaScript de forma que se volviese una representación de individuos interactiva. Un pequeño problema asociado a esta librería es la mala implementación de la misma cuando se trabaja con gráficas invisibles (que es una funcionalidad importante a la hora de diseñar la visualización de la página web).

Para la representación de los grafos, hubo que hacer mucha investigación dado que ninguna de las librerías más conocidas para dibujado de grafos permite representar grafos valorados. Una vez que se encontró la librería usada, dado que ésta solo dibujaba grafos en formato XML, se tuvo que encontrar una forma paralela de representación de los datos para asegurar que las operaciones de añadir y borrar nodos y aristas fuesen correctas y eficientes. Además, el dibujado de la solución del algoritmo ACO mediante colores para el camino mejor y opacidades y grosores de arista para las feromonas causó bastantes problemas, ya que se quería hacer de forma que no se tuviese que dibujar el grafo en cada iteración, ahorrando una gran carga de procesamiento.

Debido a la necesidad de evaluar funciones proporcionadas por el usuario se tuvo que investigar la adaptación de una librería de parser de funciones al proyecto, añadiendo funcionalidad a la misma, mediante la aplicación de los conocimientos aprendidos en Procesadores de Lenguajes.

Respecto al envío de ficheros al comparador surgieron problemas a la hora de realizar lecturas de múltiples archivos debido a que el procesamiento de ficheros por medio de JavaScript se producía de forma concurrente y fue necesario diseñar un sistema de control en la espera de las lecturas. Además, se necesitó la implementación de un sistema capaz de modificar la página web de forma dinámica manejado por el cliente en función del número de ejecuciones y tipo de las mismas. La dificultad es que se debe replicar los elementos pero modificando los nombres e identificadores, mediante un asignador de índices automático implementado mediante una expresión regular.

4.3. Estructura del proyecto

En esta sección se describe la distribución de las distintas carpetas y paquetes que conforman la estructura del proyecto. Se describen también el contenido genérico de dichas carpetas y paquetes, así como su funcionalidad.

Carpeta WEB-INF

Carpeta que contiene todo el código Java necesario para que el servidor ejecute los algoritmos, estadísticas, etc. solicitadas a través de la web.

Modelo.AlgorithmABC: Paquete encargado de la ejecución del algoritmo de optimización de la colonia de abejas (Artificial Bee Colony). Contiene la implementación de la creación de la población dados sus parámetros iniciales, así como la implementación del algoritmo iterativo ABC y los métodos auxiliares necesarios para dichas implementaciones. Código basado en [20].

Modelo.AlgorithmACO: Paquete encargado de la ejecución del algoritmo de optimización de la colonia de hormigas (Ant Colony Optimization). Contiene la implementación de la creación de la población dados sus parámetros iniciales, así como la implementación del algoritmo iterativo ACO y los métodos auxiliares necesarios para dichas implementaciones. Código basado en [21].

Modelo.AlgorithmPSO: Paquete encargado de la ejecución del algoritmo de optimización por enjambre de partículas (Particle Swarm Optimization). Contiene la implementación de la creación

de la población dados sus parámetros iniciales, así como la implementación del algoritmo iterativo PSO y los métodos auxiliares necesarios para dichas implementaciones. Contiene las clases necesarias para la elección del mejor individuo local de un vecindario dependiendo del tipo de topología utilizada. Código basado en [19].

Modelo.AlgorithmSSO: Paquete encargado de la ejecución del algoritmo de optimización de arañas sociales (Social Spider Optimization). Contiene la implementación de la creación de la población dados sus parámetros iniciales, así como la implementación del algoritmo iterativo SSO y los métodos auxiliares necesarios para dichas implementaciones. Código basado en [3].

Modelo.Base: Este paquete contiene las clases necesarias para trabajar con los distintos tipos de soluciones de cada uno de los algoritmos de optimización: por un lado soluciones a problemas de minimización de funciones en dominios, y por otro, soluciones de problemas de minimización de funciones discretas. Contiene también la clase abstracta de la que heredan todos los problemas a optimizar.

Modelo.Parser: Paquete destinado al parseo de funciones matemáticas. Es capaz de detectar errores de nomenclatura y evaluar funciones, las cuales serán introducidas por el usuario si selecciona la opción de ejecutar un problema personalizado.

Modelo.Problems: Este paquete contiene todos los problemas o funciones predefinidos que pueden ser optimizados por la aplicación.

Modelo.Statistics: Paquete destinado al cálculo de todas las operaciones estadísticas que se aplican sobre las ejecuciones solicitadas por el usuario, dándonos resultados como medias aritméticas, varianzas, etc.

Modelo.XML: Este paquete contiene funcionalidad para generar y tratar archivos XML que serán utilizados para la comunicación de los resultados entre el cliente y el servidor.

Modelo.default: Paquete que contiene las clases *Main* de Java, encargadas de crear los objetos necesarios para cada ejecución y de pasar los resultados a los controladores JSP correspondientes.

Carpeta bootstrap-3.2.0

Carpeta que contiene todos los estilos CSS y archivos JavaScript proporcionados por el *framework* de Bootstrap necesarios para la definición de los estilos en los ficheros HTML.

Carpetas en y es

Carpetas que contienen todos los ficheros HTML del proyecto. Cada carpeta corresponde a un idioma y la funcionalidad del contenido de ambas es idéntico.

Carpeta files

Carpeta que contiene todos los ficheros TSP, los cuales contienen los grafos predeterminados que se pueden cargar a la hora de ejecutar el algoritmo ACO o sus estadísticas.

Carpeta JS

Carpeta que contiene todos los ficheros JavaScript del proyecto.

amcharts: Subcarpeta que contiene toda la funcionalidad de la librería amcharts, utilizada para las representaciones gráficas del proyecto.

Carpeta JSP

Carpeta que contiene todos los ficheros JSP que sirven como controlador de la aplicación web.

5. *COMPARATIVA DE METAHEURÍSTICAS*

En los siguientes puntos se describen cada una de las funciones que son predeterminadas y que van a poder ser optimizadas (entre otras) en la aplicación.

Se incluyen, a su vez, distintas comparativas de cada uno de los algoritmos de optimización del proyecto, modificando los parámetros de cada uno de estos algoritmos de optimización, y viendo los distintos resultados que producen cada una de sus ejecuciones.

5.1. Funciones de prueba

La siguiente tabla de funciones muestra las funciones por defecto que han sido añadidas a la aplicación, así como las funciones que van a ser utilizadas para las comparativas más adelante, las cuales se utilizaron como banco de pruebas en [5].

Antes de hablar de las características que posee cada función en particular, se describen algunas propiedades de estas.

Una función es unimodal en una región si solo posee un óptimo en dicha región [34]. En estas funciones es más importante la convergencia que los propios resultados finales de la optimización, pues hay aplicaciones especialmente diseñadas para optimizar las funciones unimodales.

Por otro lado, las funciones multimodales son aquellas que poseen múltiples mínimos locales en una cierta región. En este tipo de funciones sí son importantes los mejores resultados encontrados, pues determinan la capacidad del algoritmo de optimización de escapar de los mínimos locales de dicha función.

Las funciones f_1 a f_{13} son funciones con un número elevado de dimensiones. Las funciones f_1 a f_5 son funciones unimodales. Por otro lado, la función f_6 es la función *step* o función escalón; es una función que tiene un mínimo y además es discontinua. La función f_7 es una función cuártica (o función de grado 4) con ruido. Las funciones f_8 a f_{13}son funciones multimodales en las que el número de mínimos locales aumenta exponencialmente en función del número de dimensiones de dicha función. Las funciones f_{14} a f_{23} son, en cambio, funciones con pocas dimensiones y con número reducido de mínimos locales (véase [5] para más detalles).

Una solución óptima será más difícil de encontrar cuanto mayor sea el número de dimensiones del espacio de búsqueda, así como el número de mínimos locales, que pueden provocar estancamientos en la búsqueda de mejores soluciones que las encontradas hasta el momento. Las funciones más difíciles de optimizar serán, por lo tanto, las funciones f_8 a f_{13}, pues son funciones que poseen ambas características.

Función		n	Dominio	f_{min}
Sphere Model	$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100,100]^n$	0
Schwefel's Problem 2.22	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10,10]^n$	0
Schwefel's Problem 1.2	$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} x_j \right)^2$	30	$[-100,100]^n$	0
Schwefel's Problem 2.21	$f_4(x) = max_i\{\lvert x_i \rvert, 1 \leq i \leq n\}$	30	$[-100,100]^n$	0
Generalized Rosenbrock's Function	$f_5(x) = \sum_{i=1}^{n-1} \left[100 \cdot \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	30	$[-30,30]^n$	0
Step Function	$f_6(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	30	$[-100,100]^n$	0
Quartic Function i.e. Noise	$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	$[-1.2,1.28]^n$	0
Generalized Schwefel's Problem 2.26	$f_8(x) = \sum_{i=1}^{n} -x_i sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	$[-500,500]^n$	-12569.5
Generalized Rastrigin's Function	$f_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10cos(2\pi x_i) + 10 \right]$	30	$[-5.12,5.12]^n$	0

J. Ackley's Function	$f_{10}(x)= -20esp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - exp\left(\frac{1}{n}\sqrt{\sum_{i=1}^{n}cos(2\pi x_i)}\right)+20+e$	30	$[-32,32]^n$	0
Generalized Griewank Function	$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	$[-600,600]^n$	0
Generalized Penalized Functions	$f_{12}(x)= \frac{\pi}{n}\left\{10sin^2(\pi y_i) + \sum_{i=1}^{n-1}(y_i-1)^2[1 + 10sin^2(\pi y_{i+1})](y_n-1)^2 + \right\} + \sum_{i=1}^{n}u(x_i,10,100,4),$ $y_i = 1+\frac{1}{4}(x_i+1)$ $u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i-a)^m, & x_i < -a \end{cases}$	30	$[-50,50]^n$	0
Generalized Penalized Functions	$f_{13}(x)= 0.1\left\{sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i-1)^2[1+sin^2(3\pi x_{i+1})] + (x_n-1)[1+sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n}u(x_i,5,100,4)$	30	$[-50,50]^n$	0

Shekel's Foxholes Function	$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=!}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$	2	$\begin{bmatrix}-65.536,\\ 65.536\end{bmatrix}^n$	1
Kowalik's Function	$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	$[-5,5]^n$	0.0003075
Six-Hump Camel-Back Function	$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5,5]^n$	-1.031628
Branin Function	$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cosx_1 + 10$	2	$[-5,10] \times [0,15]$	0.398
Goldstein-Price Function	$f_{18}(x) = [1\ (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [31 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2,2]^n$	3
Hartman's Family	$f_{19}(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{4} a_{ij}(x_j - p_{ij})^2\right]$	4	$[0,1]^n$	-3.86
Hartman's Family	$f_{20}(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right]$	6	$[0,1]^n$	-3.32
Shekel's Family	$f_{21}(x) = -\sum_{i=1}^{5}[(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0,10]^n$	-10
Shekel's Family	$f_{22}(x) = -\sum_{i=1}^{7}[(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0,10]^n$	-10

Shekel's Family	$f_{23}(x) = -\sum_{i=1}^{10}[(x-a_i)(x-a_i)^T + c_i]^{-1}$	4	$[0,10]^n$	-10

Algunas de las funciones a optimizar tienen tablas para determinar valores variables dependientes de distintos factores. Las tablas son mostradas a continuación:

Tabla $\boldsymbol{f_{15}}$

i	$\boldsymbol{a_i}$	$\boldsymbol{b_i^{-1}}$
1	0.1957	0.25
2	0.1947	0.5
3	0.1735	1
4	0.1600	2
5	0.0844	4
6	0.0627	6
7	0.0456	8
8	0.0342	10
9	0.0323	12
10	0.0235	14
11	0.0246	16

Tabla $\mathbf{f_{19}}$

i	$\boldsymbol{a_{ij}, \mathrm{j} = 1,2,3}$			$\mathbf{c_i}$	$\mathbf{pj = 1,2,3}$		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.03815	0.5743	0.8828

Tabla $\boldsymbol{f_{20}}$

i	$\mathbf{a_{ij}, j = 1..6}$						$\mathbf{c_i}$	$\mathbf{p_{ij}, j = 1..6}$					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

Tabla $\mathbf{f_{21}}$, $\mathbf{f_{22}}$, $\mathbf{f_{23}}$

i	$\boldsymbol{a_{ij}, \mathrm{j} = 1..4}$				$\mathbf{c_i}$
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4

5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

5.2. Comparativa de PSO

PSO permite la modificación de dos parámetros de entrada $\boldsymbol{\varphi_1}$y $\boldsymbol{\varphi_2}$, que modifican la influencia local y global, respectivamente, que ejercen los demás individuos de la población sobre la velocidad de dicho individuo, siguiendo la fórmula dada por [1]

$$\boldsymbol{v^{k+1} = \omega \cdot v^k + \varphi_1 \cdot R1 \cdot (pBest - x^k) + \varphi_2 \cdot R2 \cdot (g - x^k)}$$

donde $\boldsymbol{\omega}$ determina la importancia que tiene la inercia del individuo antes de modificar su velocidad.

Se muestran los resultados obtenidos para algunas funciones modificando los parámetros $\boldsymbol{\varphi_1}$y $\boldsymbol{\varphi_2}$, que permiten modificar la importancia del factor local y global a la hora de determinar la nueva velocidad del individuo.

Para las siguientes comparativas se han realizado 50 ejecuciones con poblaciones de 30 individuos y un número de iteraciones de 1000, donde **BF** es el *mejor fitness* encontrado en las ejecuciones realizadas, **AF** es la *media de los mejores fitness* calculados en cada una de las ejecuciones realizadas y **VF** es la *varianza de los mejores fitness* calculados en cada una de las ejecuciones realizadas.

El valor escogido para $\boldsymbol{\omega}$ en esta comparativa, es un valor dependiente del número de iteraciones máximo $maxIter$, de la iteración actual $iter$, así como de de dos factores ω_i y ω_f, que indican el valor inercial inicial y final de cada individuo, respectivamente. En la primera iteración, el valor de $\omega = \omega_i$. Para el cálculo de $\boldsymbol{\omega}$ en el resto de iteraciones, se sigue la siguiente fórmula dada por [6]:

$$(\omega - \omega_f) * \frac{(maxIter - iter)}{maxIter} + \omega_f$$

Se escogen valores de $\boldsymbol{\varphi_1 = \varphi_2} = 2$ que son buenos valores de los parámetros según [1] y de $\boldsymbol{\varphi_1 = \varphi_2} = 2.89$. Estos valores han sido calculados experimentalmente, probando múltiples y distintos valores para comprobar el comportamiento de las funciones, y llegando a la conclusión de que producen mejores resultados en las funciones estudiadas que el valor de los parámetros recomendado por [6], $\boldsymbol{\varphi_1 = \varphi_2} = 2$. Los factores inerciales inicial ω_i y final ω_f toman el valor 0.9 y 0.4, respectivamente según [21].

Se marcan en negrita los mejores resultados de **BF** y **AF** (en caso de tener en dos o más algoritmos el mismo valor de **BF** o **AF,** se marcarán todos los mejores resultados).

PSO Global

A continuación se muestra la tabla con los resultados tras ejecutar PSO global con el ajuste de parámetros citado anteriormente

		φ_1= 2	φ_2=2	φ_1= 2.89	φ_2=2.89
f_1	BF	397.8702		**0.0044**	
	AF	2443.4815		**0.2914**	
	VF	2298874.0787		0.0822	
f_2	BF	9.5528		**0.0826**	
	AF	25.2312		**4.7333**	
	VF	261.7418		58.4657	
f_3	BF	**0.0**		**0.0**	
	AF	**0.0**		**0.0**	
	VF	0.0		0.0	
f_4	BF	13.2182		**0.0108**	
	AF	19.4784		**0.1959**	
	VF	19.1846		0.0105	
f_5	BF	17789.1234		**29.0095**	
	AF	343192.9554		**1845.3625**	
	VF	1.0274E11		1.6195E8	
f_6	BF	548.0		**0.0**	
	AF	2115.42		**196.12**	
	VF	904918.4935		1921152.1486	
f_7	BF	0.0869		**0.0016**	
	AF	1.0099		**0.1162**	
	VF	1.527		0.2813	
f_8	BF	-8277.2765		**-10003.9225**	
	AF	-6633.3259		**-7258.8074**	
	VF	779020.7655		2171255.0338	
f_9	BF	94.2497		**0.061**	
	AF	155.9898		**62.1462**	
	VF	881.4871		1511.5305	
f_{10}	BF	7.1717		**0.0542**	
	AF	11.0619		**0.2411**	
	VF	3.7058		0.0383	
f_{11}	BF	4.5987		**0.0521**	
	AF	21.0628		**0.4551**	
	VF	147.0574		0.0626	
f_{12}	BF	9.4954		**0.0424**	
	AF	273.8798		**0.183**	
	VF	2280437.2292		0.0091	
f_{13}	BF	39.5917		**0.3854**	
	AF	56599.9305		**1.529**	
	VF	1.0533E10		0.319	

f_{14}	BF	**0.998**	**0.998**
	AF	**2.9076**	6.3098
	VF	7.6441	26.6627
f_{15}	BF	**3.0E-4**	8.0E-4
	AF	**0.0088**	0.0089
	VF	1.0E-4	1.0E-4
f_{16}	BF	**-1.0316**	**-1.0316**
	AF	**-1.0316**	**-1.0316**
	VF	0.0	0.0
f_{17}	BF	**0.3979**	**0.3979**
	AF	**0.3979**	**0.3979**
	VF	0.0	0.0
f_{18}	BF	**3.0**	**3.0**
	AF	**3.0**	**3.0**
	VF	0.0	0.0
f_{19}	BF	**-3.8628**	**-3.8628**
	AF	**-3.8612**	-3.8531
	VF	0.0	0.0024
f_{20}	BF	**-3.322**	**-3.322**
	AF	**-3.1647**	-2.9626
	VF	0.0653	0.1827
f_{21}	BF	**-10.1532**	**-10.1532**
	AF	-6.6293	**-9.7239**
	VF	9.2546	2.7015
f_{22}	BF	**-10.4029**	**-10.4029**
	AF	-7.4295	**-10.3945**
	VF	12.4424	0.0035
f_{23}	BF	**-10.5364**	**-10.5364**
	AF	-7.7542	**-10.3817**
	VF	13.3046	1.195

PSO Local

A continuación se muestra la tabla con los resultados tras ejecutar PSO local con el ajuste de parámetros citado anteriormente

		φ_1= 2	φ_2=2	φ_1= 2.89	φ_2=2.89
f_1	BF	900.2303		**0.1091**	
	AF	2128.2234		**3.4909**	
	VF	363876.6105		10.9356	
f_2	BF	15.8282		**0.2213**	
	AF	21.9689		**1.3812**	
	VF	16.196		0.4649	

f_3	BF	**0.0**	**0.0**
	AF	**0.0**	**0.0**
	VF	0.0	0.0
f_4	BF	13.0958	**0.0656**
	AF	17.9972	**0.5163**
	VF	9.3996	0.1012
f_5	BF	72296.633	**30.0745**
	AF	404742.4693	**54.1765**
	VF	9.6954E10	541.7625
f_6	BF	523.0	**0.0**
	AF	2122.94	**0.9**
	VF	428699.1188	2.1327
f_7	BF	0.074	**0.0013**
	AF	0.4095	**0.0099**
	VF	0.0586	0.0
f_8	BF	**-9534.1112**	-9201.4637
	AF	**-7681.4576**	-7549.1674
	VF	802013.3071	623869.8123
f_9	BF	110.1376	**3.0241**
	AF	144.5403	**53.3959**
	VF	356.2049	630.7573
f_{10}	BF	4.6363	**0.1384**
	AF	10.1452	**1.9692**
	VF	2.338	20.168
f_{11}	BF	10.3138	**0.1337**
	AF	20.4148	**0.8592**
	VF	51.3336	0.0821
f_{12}	BF	7.9812	**0.2188**
	AF	109.0835	**0.5153**
	VF	381067.073	0.0199
f_{13}	BF	16.3314	**1.8113**
	AF	55424.8742	**2.9604**
	VF	8.4734E9	0.2475
f_{14}	BF	1.992	**1.4577**
	AF	490.0398	**94.6995**
	VF	4960.2387	22287.7724
f_{15}	BF	**3.0E-4**	**3.0E-4**
	AF	**0.0013**	0.0021
	VF	0.0	0.0
f_{16}	BF	**-1.0316**	**-1.0316**
	AF	**-1.0316**	**-1.0316**
	VF	0.0	0.0
f_{17}	BF	**0.3979**	**0.3979**
	AF	**0.3979**	**0.3979**
	VF	0.0	0.0

f_{18}	BF	**3.0**	**3.0**
	AF	**3.0**	**3.0**
	VF	0.0	0.0
f_{19}	BF	**-3.8628**	**-3.8628**
	AF	**-3.8628**	**-3.8628**
	VF	0.0	0.0
f_{20}	BF	**-3.322**	-3.3219
	AF	-3.297	**-3.3032**
	VF	0.0024	0.0019
f_{21}	BF	**-10.1532**	-10.153
	AF	-10.0522	**-10.1486**
	VF	0.5105	0.0
f_{22}	BF	**-10.4029**	-10.4027
	AF	-9.6067	**-10.397**
	VF	4.0299	0.0
f_{23}	BF	**-10.5364**	-10.5363
	AF	-9.8386	**-10.5318**
	VF	3.6922	1.0E-4

PSO Von Neumman

A continuación se muestra la tabla con los resultados tras ejecutar PSO con la topología Von Neumman con el ajuste de parámetros citado anteriormente

		φ_1= 2	φ_2=2	φ_1= 2.89	φ_2=2.89
f_1	BF	620.6416		**0.0418**	
	AF	1988.7561		**2.0706**	
	VF	692608.3529		2.5616	
f_2	BF	13.2002		**0.3208**	
	AF	21.6411		**0.8442**	
	VF	20.4486		0.0887	
f_3	BF	**0.0**		**0.0**	
	AF	**0.0**		**0.0**	
	VF	0.0		0.0	
f_4	BF	11.0811		**0.0417**	
	AF	17.6438		**0.5279**	
	VF	12.2626		0.0604	
f_5	BF	49665.8962		**30.6672**	
	AF	358336.8245		**69.5207**	
	VF	6.4174E10		22956.4637	
f_6	BF	790.0		**0.0**	
	AF	2113.98		**0.6**	
	VF	630179.1629		0.6531	

f_7	BF	0.0904	**0.0011**
	AF	0.5187	**0.0115**
	VF	0.1129	0.0
f_8	BF	-8613.1208	**-9548.5557**
	AF	**-7315.6431**	-6993.4539
	VF	530128.9947	1527942.966
f_9	BF	104.5167	**3.4278**
	AF	152.209	**49.1614**
	VF	287.1144	802.4006
f_{10}	BF	6.0984	**0.2741**
	AF	10.0838	**2.9069**
	VF	1.5862	33.7756
f_{11}	BF	9.486	**0.2818**
	AF	21.2434	**0.92**
	VF	55.8228	0.034
f_{12}	BF	5.7272	**0.1773**
	AF	29.0861	**0.4043**
	VF	2386.7894	0.0194
f_{13}	BF	33.6107	**1.5073**
	AF	85484.2947	**2.477**
	VF	2.5698E10	0.1756
f_{14}	BF	**0.998**	1.04
	AF	**2.1848**	43.5823
	VF	2.8232	9355.268
f_{15}	BF	**3.0E-4**	**3.0E-4**
	AF	**0.0017**	0.0031
	VF	0.0	0.0
f_{16}	BF	**-1.0316**	**-1.0316**
	AF	**-1.0316**	**-1.0316**
	VF	0.0	0.0
f_{17}	BF	**0.3979**	**0.3979**
	AF	**0.3979**	**0.3979**
	VF	0.0	0.0
f_{18}	BF	**3.0**	**3.0**
	AF	**3.0**	**3.0**
	VF	0.0	0.0
f_{19}	BF	**-3.8628**	**-3.8628**
	AF	**-3.8628**	**-3.8628**
	VF	0.0	0.0
f_{20}	BF	**-3.322**	-3.3219
	AF	**-3.2614**	-3.2584
	VF	0.0042	0.005
f_{21}	BF	**-10.1532**	-10.1531
	AF	-9.5987	**-9.9657**
	VF	2.9184	1.7187

f_{22}	BF	**-10.4029**	**-10.4029**
	AF	-9.8177	**-10.4004**
	VF	3.1919	0.0
f_{23}	BF	-10.5364	**-10.5363**
	AF	-9.8997	**-10.534**
	VF	3.8015	0.0

Se puede observar como el uso del valor calculado experimentalmente para $\boldsymbol{\varphi_1}$ y $\boldsymbol{\varphi}_2$, 2.89, produce buenos resultados en casi todos los casos, pues da algo más de importancia a los factores sociales que $\boldsymbol{\varphi_1} = \boldsymbol{\varphi}_2 = 2$, permitiendo que el movimiento del individuo no esté tan determinado por la inercia que lleva debido a la velocidad actual.

Comparativa de distintas topologías de PSO

Se realiza una comparativa entre las distintas topologías implementadas para PSO, utilizando el valor experimental calculando anteriormente 2.89, y dos nuevas variantes, $\boldsymbol{\varphi_1} = 1, \boldsymbol{\varphi_2} = 4$ y $\boldsymbol{\varphi_1} = 4, \boldsymbol{\varphi_2} = 1$. Se utilizan dichos valores para calcular el cambio que produce el dar más importancia al factor local o global en alguna de las funciones estudiadas.

Se marcan en negrita los mejores resultados de **BF** y **AF** (en caso de tener en dos o más algoritmos el mismo valor de **BF** o **AF,** se marcarán todos los mejores resultados).

			$\boldsymbol{\varphi_1} = 4$ $\boldsymbol{\varphi_2} = 1$	$\boldsymbol{\varphi_1} = 1$ $\boldsymbol{\varphi_2} = 4$	$\boldsymbol{\varphi_2} = 2.89$ $\boldsymbol{\varphi_2} = 2.89$
f_1	Global	BF	**5.0E-4**	**5.0E-4**	0.0044
		AF	0.2048	**0.0593**	0.2914
		VF	0.0429	0.0036	0.0822
	Local	BF	0.0111	**5.0E-4**	0.1091
		AF	1.1602	0.996	3.4909
		VF	1.4744	0.9167	10.9356
	Von Neumann	BF	0.0034	0.0038	0.0418
		AF	0.378	0.4708	2.0706
		VF	0.1432	0.2107	2.5616
f_2	Global	BF	**0.0324**	0.0528	0.0826
		AF	0.4987	3.1958	4.7333
		VF	1.9988	33.6574	58.4657
	Local	BF	0.0952	0.0859	0.2213
		AF	0.6694	0.5679	1.3812
		VF	0.1964	0.1333	0.4649
	Von Neumann	BF	0.0358	0.0872	0.3208
		AF	0.3467	**0.3371**	0.8442
		VF	0.0446	0.0308	0.0887

f_3	Global	BF	**0.0**	**0.0**	**0.0**
		AF	**0.0**	**0.0**	**0.0**
		VF	0.0	0.0	0.0
	Local	BF	**0.0**	**0.0**	**0.0**
		AF	**0.0**	**0.0**	**0.0**
		VF	0.0	0.0	0.0
	Von Neumann	BF	**0.0**	**0.0**	**0.0**
		AF	**0.0**	**0.0**	**0.0**
		VF	0.0	0.0	0.0
f_4	Global	BF	0.0214	**0.006**	0.0108
		AF	0.1408	**0.0846**	0.1959
		VF	0.0083	0.0041	0.0105
	Local	BF	0.032	0.0377	0.0656
		AF	0.3677	0.3121	0.5163
		VF	0.0542	0.0276	0.1012
	Von Neumann	BF	0.0261	0.0321	0.0417
		AF	0.1984	0.2257	0.5279
		VF	0.0129	0.0182	0.0604
f_5	Global	BF	**28.6642**	28.7713	29.0095
		AF	44.9889	44.555	1845.3625
		VF	3991.6607	4134.8061	1.6195E8
	Local	BF	29.3152	29.022	30.0745
		AF	40.3442	37.9004	54.1765
		VF	154.3091	228.8659	541.7625
	Von Neumann	BF	29.0385	29.0009	30.6672
		AF	**32.8371**	33.1888	69.5207
		VF	26.8852	10.7737	22956.4637
f_6	Global	BF	**0.0**	**0.0**	**0.0**
		AF	**0.0**	200.02	196.12
		VF	0.0	1999991.8567	1921152.1486
	Local	BF	**0.0**	**0.0**	**0.0**
		AF	0.26	0.1	0.9
		VF	0.3596	0.0918	2.1327
	Von Neumann	BF	**0.0**	**0.0**	**0.0**
		AF	0.02	0.08	0.6
		VF	0.02	0.0751	0.6531
f_7	Global	BF	6.0E-4	3.0E-4	0.0016
		AF	**0.0044**	0.5397	0.1162
		VF	0.0	3.5266	0.2813
	Local	BF	3.0E-4	**7.0E-4**	0.0013
		AF	0.0063	**0.0044**	0.0099
		VF	0.0	0.0	0.0
	Von Neumann	BF	3.0E-4	5.0E-4	0.0011
		AF	0.0053	0.0046	0.0115
		VF	0.0	0.0	0.0

f_8	Global	BF	-10490.208	**-11067.4459**	-10003.9225
		AF	-8301.5404	-6923.611	-7258.8074
		VF	1536175.9265	3261368.5203	2171255.0338
	Local	BF	-9882.6117	-10513.4517	-9201.4637
		AF	-8169.2122	-7988.9521	-7549.1674
		VF	519305.6457	1455202.4179	623869.8123
	Von Neumann	BF	-10420.0259	-10171.9536	-9548.5557
		AF	**-8437.1169**	-6425.1929	-6993.4539
		VF	860808.007	1995327.3564	1527942.966
f_9	Global	BF	2.0985	16.783	0.061
		AF	55.8169	83.8448	62.1462
		VF	883.7234	1609.0997	1511.5305
	Local	BF	1.8786	4.2491	3.0241
		AF	36.9833	39.0745	53.3959
		VF	396.9985	552.4164	630.7573
	Von Neumann	BF	**0.0286**	1.8469	3.4278
		AF	29.4906	**28.7609**	49.1614
		VF	417.8045	295.5824	802.4006
f_{10}	Global	BF	**0.0063**	0.0142	0.0542
		AF	**0.1789**	2.6689	0.2411
		VF	0.0412	37.2493	0.0383
	Local	BF	0.0238	0.0206	0.1384
		AF	0.5611	5.248	1.9692
		VF	0.1919	61.4979	20.168
	Von Neumann	BF	0.041	0.0587	0.2741
		AF	0.2489	2.3516	2.9069
		VF	0.0351	32.3338	33.7756
f_{11}	Global	BF	**0.0029**	0.0039	0.0521
		AF	0.3615	**0.2163**	0.4551
		VF	0.0783	0.0305	0.0626
	Local	BF	0.0568	0.0508	0.1337
		AF	0.6991	0.6778	0.8592
		VF	0.0935	0.0845	0.0821
	Von Neumann	BF	0.0024	0.0278	0.2818
		AF	0.489	0.5474	0.92
		VF	0.0867	0.0666	0.034
f_{12}	Global	BF	0.0398	**0.0023**	0.0424
		AF	0.2428	0.1695	0.183
		VF	0.0695	0.0523	0.0091
	Local	BF	0.1451	0.1112	0.2188
		AF	0.3606	0.2837	0.5153
		VF	0.0315	0.0104	0.0199
	Von Neumann	BF	0.0601	0.0414	0.1773
		AF	0.2013	**0.1683**	0.4043
		VF	0.0187	0.0052	0.0194

f_{13}	Global	BF	0.574	**0.241**	0.3854
		AF	1.6308	**1.427**	1.529
		VF	0.2362	0.3501	0.319
	Local	BF	1.4003	1.1729	1.8113
		AF	2.2737	2.1007	2.9604
		VF	0.2123	0.2347	0.2475
	Von Neumann	BF	0.751	0.4214	1.5073
		AF	1.4876	1.4378	2.477
		VF	0.1567	0.231	0.1756
f_{14}	Global	BF	**0.998**	**0.998**	**0.998**
		AF	1.8368	**1.4828**	6.3098
		VF	4.3683	1.0439	26.6627
	Local	BF	**0.998**	**0.998**	1.4577
		AF	64.8727	17.8661	94.6995
		VF	24328.3901	5808.4158	22287.7724
	Von Neumann	BF	0.998	0.998	1.04
		AF	1.7014	1.8749	43.5823
		VF	1.0047	1.3098	9355.268
f_{15}	Global	BF	3.0E-4	3.0E-4	**8.0E-4**
		AF	0.0032	0.0097	0.0089
		VF	0.0	1.0E-4	1.0E-4
	Local	BF	3.0E-4	3.0E-4	3.0E-4
		AF	0.0014	0.0021	0.0021
		VF	0.0	0.0	0.0
	Von Neumann	BF	3.0E-4	3.0E-4	3.0E-4
		AF	**9.0E-4**	0.0038	0.0031
		VF	0.0	0.0	0.0
f_{16}	Global	BF	**-1.0316**	**-1.0316**	**-1.0316**
		AF	**-1.0316**	**-1.0316**	**-1.0316**
		VF	0.0	0.0	0.0
	Local	BF	**-1.0316**	**-1.0316**	**-1.0316**
		AF	**-1.0316**	**-1.0316**	**-1.0316**
		VF	0.0	0.0	0.0
	Von Neumann	BF	**-1.0316**	**-1.0316**	**-1.0316**
		AF	**-1.0316**	**-1.0316**	**-1.0316**
		VF	0.0	0.0	0.0
f_{17}	Global	BF	**0.3979**	**0.3979**	**0.3979**
		AF	**0.3979**	**0.4288**	**0.3979**
		VF	0.0	0.0478	0.0
	Local	BF	**0.3979**	**0.3979**	**0.3979**
		AF	**0.3979**	**0.3979**	**0.3979**
		VF	0.0	0.0	0.0
	Von Neumann	BF	**0.3979**	**0.3979**	**0.3979**
		AF	**0.3979**	**0.3979**	**0.3979**
		VF	0.0	0.0	0.0

f_{18}	Global	BF	**3.0**	**3.0**	**3.0**
		AF	**3.0**	8.0164	**3.0**
		VF	0.0	403.1118	0.0
	Local	BF	**3.0**	**3.0**	**3.0**
		AF	**3.0**	**3.0**	**3.0**
		VF	0.0	0.0	0.0
	Von Neumann	BF	**3.0**	**3.0**	**3.0**
		AF	**3.0**	**3.0**	**3.0**
		VF	0.0	0.0	0.0
f_{19}	Global	BF	**-3.8628**	**-3.8628**	**-3.8628**
		AF	**-3.8628**	-3.8468	-3.8531
		VF	0.0	0.0047	0.0024
	Local	BF	**-3.8628**	**-3.8628**	**-3.8628**
		AF	**-3.8628**	**-3.8628**	**-3.8628**
		VF	0.0	0.0	0.0
	Von Neumann	BF	**-3.8628**	**-3.8628**	**-3.8628**
		AF	**-3.8628**	**-3.8628**	**-3.8628**
		VF	0.0	0.0	0.0
f_{20}	Global	BF	**-3.322**	**-3.322**	**-3.322**
		AF	-3.2584	-3.0279	-2.9626
		VF	0.0039	0.1686	0.1827
	Local	BF	**-3.322**	**-3.322**	-3.3219
		AF	**-3.3174**	-3.2994	-3.3032
		VF	4.0E-4	0.0022	0.0019
	Von Neumann	BF	**-3.322**	**-3.322**	-3.3219
		AF	-3.2849	-3.211	-3.2584
		VF	0.0032	0.057	0.005
f_{21}	Global	BF	**-10.1532**	**-10.1532**	**-10.1532**
		AF	-9.8523	-10.0027	-9.7239
		VF	2.2175	1.1318	2.7015
	Local	BF	**-10.1532**	**-10.1532**	**-10.153**
		AF	-10.1528	-10.1531	-10.1486
		VF	0.0	0.0	0.0
	Von Neumann	BF	**-10.1532**	**-10.1532**	-10.1531
		AF	**-10.1532**	**-10.1532**	-9.9657
		VF	0.0	0.0	1.7187
f_{22}	Global	BF	**-10.4029**	**-10.4029**	**-10.4029**
		AF	**-10.4029**	-9.9076	-10.3945
		VF	0.0	3.9344	0.0035
	Local	BF	**-10.4029**	**-10.4029**	-10.4027
		AF	-10.4027	-10.4027	-10.397
		VF	0.0	0.0	0.0
	Von Neumann	BF	**-10.4029**	**-10.4029**	**-10.4029**
		AF	**-10.4029**	**-10.4029**	-10.4004
		VF	0.0	0.0	0.0

f_{23}	Global	BF	**-10.5364**	**-10.5364**	**-10.5364**
		AF	**-10.5364**	**-10.5364**	-10.3817
		VF	0.0	0.0	1.195
	Local	BF	**-10.5364**	**-10.5364**	-10.5363
		AF	-10.5363	-10.5361	-10.5318
		VF	0.0	0.0	1.0E-4
	Von Neumann	BF	**-10.5364**	**-10.5364**	-10.5363
		AF	**-10.5364**	**-10.5364**	-10.534
		VF	0.0	0.0	0.0

Se comprueba que la topología que mejor funciona para las funciones estudiadas es la topología PSO global, pues tiene en cuenta toda la población para la búsqueda del mejor valor, determinando su movimiento de atracción hacia dicha posición, y facilitando así la búsqueda de la mejor solución.

Por otro lado, PSO local tiene en cuenta los vecinos creados inmediatamente antes y después que el individuo actual, provocando que solo tenga como referencia dos individuos, que normalmente no serán los mejores valores de los que recibir influencia a la hora de determinar su movimiento.

PSO Von Neumman en cambio, a pesar de funcionar como PSO local (determina el vecindario por orden de creación), posee un mayor número de vecinos, lo cual aumenta la probabilidad de dar con posiciones del espacio de búsqueda mejores a las que ser atraído.

5.3. Comparativa de SSO

El algoritmo que implementa el comportamiento de las arañas sociales modifica la posición de las arañas hembra siguiendo la fórmula

$$f_i^{k+1} = \begin{cases} f_i^k + \alpha \cdot Vibc_i \cdot \left(c - f_i^k\right) + \beta \cdot Vibb_i \cdot \left(b - f_i^k\right) + \delta \cdot \left(rand - \frac{1}{2}\right), & si\ r_m < PF \\ f_i^k - \alpha \cdot Vibc_i \cdot \left(c - f_i^k\right) - \beta \cdot Vibb_i \cdot \left(b - f_i^k\right) + \delta \cdot (rand - \frac{1}{2}), & si\ r_m \geq PF \end{cases}$$

Donde los parámetros $\alpha, \beta, \delta, rand\ y\ r_m$ son parámetros totalmente aleatorios.

PF es un parámetro de entrada que decide la probabilidad que tiene una araña hembra de ser atraída o repelida hacia las arañas $c\ y\ b$, la araña más cercana a f y con un peso mayor que el suyo y la araña con mayor peso de todas, respectivamente.

Por otro lado, los machos dominantes son atraídos hacia las arañas hembra, con mayor o menor fuerza, dependiendo también de factores aleatorios, como son $\alpha, \delta\ y\ rand$ en función de la siguiente ecuación

$$m_i^{k+1} = m_i^k + \alpha \cdot Vibf_i \cdot (f - m_i^k) + \delta \cdot \left(rand - \tfrac{1}{2}\right)$$

Por lo tanto, y teniendo como único parámetro de entrada para el usuario el parámetro PF, y donde la mayor parte de las decisiones se toman mediante parámetros aleatorios, el comportamiento de este algoritmo es totalmente imprevisible, y en ciertas ocasiones nada bueno.

Se muestran los resultados obtenidos para algunas funciones modificando el parámetro PF, con una población de 30 individuos y un número de iteraciones de 1000, donde **BF** es el *mejor fitness* encontrado en las ejecuciones realizadas, **AF** es la *media de los mejores fitness* calculados en cada una de las ejecuciones realizadas y **VF** es la *varianza de los mejores fitness* calculados en cada una de las ejecuciones realizadas.

Se dan valores a PF de 0.7, que produce buenos resultados, según [3], de 0.2 y 0.9 para ver el comportamiento del algoritmo antes una atracción y una repulsión muy fuerte hacia los mejores individuos, y un valor de 0.6, que tras varias pruebas experimentales se comprueba que produce unos buenos resultados.

Se marcan en negrita los mejores resultados de **BF** y **AF** (en caso de tener en dos o más algoritmos el mismo valor de **BF** o **AF,** se marcarán todos los mejores resultados).

		$p_f = 0.2$	$p_f = 0.6$	$p_f = 0.7$	$p_f = 0.9$
f_1	BF	**9234.8401**	9819.4582	14000.5676	10468.2903
	AF	**9362.2625**	10018.6706	14993.6971	10811.292
	VF	126073.6161	67384.0215	4584870.1148	138522.7905
f_2	BF	58.1392	22.3473	22.0684	**20.1837**
	AF	61.3486	24.943	23.1058	**21.2426**
	VF	515.0087	10.1331	10.6005	4.618
f_3	BF	**0.0**	**0.0**	**0.0**	**0.0**
	AF	2.0E-4	**3.0E-4**	0.0	0.0014
	VF	0.0	0.0	0.0	0.0
f_4	BF	32.42	69.1463	**29.1916**	33.728
	AF	34.5618	72.3469	**29.5538**	35.7813
	VF	24.4832	2.5904	3.2118	7.1987
f_5	BF	5.5866E7	**2098356.9573**	5786481.6688	1.6897E7
	AF	5.7414E7	**2372079.7418**	8148564.6842	1.7615E7
	VF	1.6530E13	9.6499E11	1.3735E13	5.6647E12
f_6	BF	**9171.0**	27848.0	12856.0	9895.0
	AF	**9819.86**	29298.38	14772.28	10091.36
	VF	132239.6331	2501415.5873	2400339.1853	778892.6024
f_7	BF	1.1817	**0.2394**	0.6537	0.7338
	AF	1.9279	**0.2819**	0.8004	0.9658
	VF	4.9683	0.0154	0.1058	0.0182
f_8	BF	-2262.0353	**-2783.6866**	-2299.0978	-2481.1339
	AF	-2262.0338	**-2756.0118**	-2278.9265	-2327.6902
	VF	1.0E-4	4054.1175	5779.0978	54068.4037

f_9	BF	216.4577	173.6945	170.5841	**165.1552**
	AF	222.5438	183.0268	174.9192	**172.1954**
	VF	233.193	77.7493	146.9243	124.815
f_{10}	BF	15.877	**15.563**	16.2702	15.9429
	AF	15.9527	**15.6955**	16.5216	16.0339
	VF	0.0211	0.0208	0.0395	0.0619
f_{11}	BF	**60.8146**	75.3571	221.2147	96.4135
	AF	**64.425**	82.0991	227.3075	100.2964
	VF	98.1735	65.971	44.1524	30.2201
f_{12}	BF	1.1451E7	6560946.9788	**5456744.9598**	1543801.4006
	AF	1.5077E7	6671926.3937	7142411.9631	**2169031.204**
	VF	3.9814E13	1.1311E11	2.9566E13	2.7496E12
f_{13}	BF	1.5563E7	2.3337E7	**3954904.9488**	1.6003E7
	AF	1.6445E7	2.6291E7	**6743343.7532**	2.1595E7
	VF	5.8862E12	1.5215E14	1.9492E13	2.4391E13
f_{14}	BF	**500.0**	**500.0**	**500.0**	**500.0**
	AF	**500.0**	**500.0**	**500.0**	**500.0**
	VF	0.0	0.0	0.0	0.0
f_{15}	BF	0.0014	0.0014	0.0016	**4.0E-4**
	AF	0.0019	0.0014	0.0017	**5.0E-4**
	VF	0.0	0.0	0.0	0.0
f_{16}	BF	-1.0314	**-1.0316**	**-1.0316**	**-1.0316**
	AF	-1.0257	**-1.0316**	**-1.0316**	**-1.0316**
	VF	6.0E-4	0.0	0.0	0.0
f_{17}	BF	0.3984	**0.3979**	**0.3979**	**0.3979**
	AF	0.4016	**0.3979**	**0.398**	**0.3979**
	VF	1.0E-4	0.0	0.0	0.0
f_{18}	BF	3.0655	**3.0**	**3.0**	**3.0**
	AF	3.1925	**3.0**	**3.0**	**3.0**
	VF	0.1228	0.0	0.0	0.0
f_{19}	BF	-3.8177	-3.8593	**-3.862**	**-3.862**
	AF	-3.8045	-3.8559	-3.8581	**-3.862**
	VF	0.0011	0.0	1.0E-4	0.0
f_{20}	BF	-2.6005	-3.1816	-3.2134	**-3.2618**
	AF	-2.519	-3.1532	-3.1939	**-3.2376**
	VF	0.0093	0.0021	0.0034	0.0014
f_{21}	BF	-5.5081	**-10.1305**	-9.9315	-10.0478
	AF	-5.0019	-9.9728	-9.1539	**-10.021**
	VF	0.5291	0.0303	3.9165	0.0014
f_{22}	BF	-5.6248	-10.3576	**-10.3612**	-10.3606
	AF	-4.0057	**-10.2467**	-10.2188	-10.0173
	VF	1.2551	0.0768	0.889	1.6975
f_{23}	BF	-9.6352	**-10.4781**	-10.2955	-10.4737
	AF	-8.2954	-10.3808	-10.0925	**-10.3913**
	VF	3.8058	0.0913	0.8789	0.0577

Debido a la atracción a los individuos con un peso mayor, se produce una cierta mejora global en la media de la población. Por otro lado, la repulsión a ciertos individuos conduce a explorar el resto del espacio de búsqueda, impidiendo el estancamiento en mínimos locales, y pudiendo encontrarse soluciones mejores de las encontradas hasta el momento.

Por ello, un valor de PF entorno a 0.6 produce buenos resultados, ya que provoca dichos comportamientos de forma más o menos equitativa, dando una probabilidad similar tanto al hecho de buscar nuevas soluciones en el espacio de búsqueda, así como al de ser atraído a mejores posiciones comparadas con la del individuo actual, sin dar lugar a un número excesivamente grande de búsquedas que no produzcan nuevos buenos resultados.

5.4. Comparativa de ABC

El algoritmo de la colonia de abejas permite la modificación de un parámetro de entrada definido por el usuario $limit$, que permite decidir el número de intentos que tiene una abeja de modificar la posición de la fuente de alimento que está explotando antes de descartarla y buscar otra totalmente aleatoria.

Se muestran los resultados obtenidos para algunas funciones modificando el parámetro $limit$, con una población de 30 individuos y un número de iteraciones de 1000, donde **BF** es el *mejor fitness* encontrado en las ejecuciones realizadas, **AF** es la *media de los mejores fitness* calculados en cada una de las ejecuciones realizadas y **VF** es la *varianza de los mejores fitness* calculados en cada una de las ejecuciones realizadas.

El parámetro $limit$ más utilizado determina su valor como $limit = N * D$, siendo D el número de dimensiones de la función a optimizar y N el número de individuos de la población (en este caso, $limit = 30 * D$) [13].

Se introducen dos nuevos valores para dicho parámetro $limit$, para comparar su comportamiento.

Se marcan en negrita los mejores resultados de **BF** y **AF** (en caso de tener en dos o más algoritmos el mismo valor de **BF** o **AF,** se marcarán todos los mejores resultados).

		$\boldsymbol{limit} = 20$	$\boldsymbol{limit} = N * D$	$\boldsymbol{limit} = 1000$
f_1	BF	**0.0**	**0.0**	**0.0**
	AF	0.0305	**0.0**	**0.0**
	VF	0.005	0.0	0.0
f_2	BF	**0.0**	**0.0**	**0.0**
	AF	0.012	**0.0**	**0.0**
	VF	2.0E-4	0.0	0.0
f_3	BF	**0.0**	**0.0**	**0.0**
	AF	1.0E-4	**0.0**	**0.0**
	VF	0.0	0.0	0.0

f_4	BF	60.9324	**10.682**	13.6186
	AF	70.3057	**23.0761**	23.316
	VF	13.6296	33.249	17.055
f_5	BF	26.725	0.1495	**0.1001**
	AF	217.784	**3.6384**	4.7734
	VF	11291.0289	11.5164	13.8516
f_6	BF	4.0	**0.0**	**0.0**
	AF	30.38	**0.0**	**0.0**
	VF	409.7914	0.0	0.0
f_7	BF	0.4663	0.0717	**0.0667**
	AF	1.0157	**0.1389**	0.1407
	VF	0.0631	0.0016	0.0015
f_8	BF	-11488.4145	-12327.8149	**-12332.5413**
	AF	-10288.1813	-12042.1991	**-12056.661**
	VF	150346.3848	22440.2087	24149.0273
f_9	BF	16.7796	**0.0**	**0.0**
	AF	29.1137	**0.7027**	0.7818
	VF	45.9155	0.5685	0.4855
f_{10}	BF	0.6137	**0.0**	**0.0**
	AF	2.0986	**1.0E-4**	**1.0E-4**
	VF	0.5065	0.0	0.0
f_{11}	BF	1.0E-4	**0.0**	**0.0**
	AF	0.0897	0.0056	**0.0055**
	VF	0.0058	2.0E-4	1.0E-4
f_{12}	BF	0.0	**0.0**	**0.0**
	AF	0.0037	**0.0**	**0.0**
	VF	0.0	0.0	0.0
f_{13}	BF	0.0	**0.0**	**0.0**
	AF	0.0286	**0.0**	**0.0**
	VF	0.0012	0.0	0.0
f_{14}	BF	14.8775	**0.998**	0.998
	AF	441.2962	**37.287**	244.2164
	VF	19011.928	13536.0979	58350.2798
f_{15}	BF	**5.0E-4**	4.0E-4	4.0E-4
	AF	**9.0E-4**	7.0E-4	8.0E-4
	VF	0.0	0.0	0.0
f_{16}	BF	**-1.0316**	**-1.0316**	**-1.0316**
	AF	**-1.0316**	**-1.0316**	**-1.0316**
	VF	0.0	0.0	0.0
f_{17}	BF	**0.3979**	**0.3979**	**0.3979**
	AF	**0.3979**	**0.3979**	**0.3979**
	VF	0.0	0.0	0.0
f_{18}	BF	**3.0**	**3.0**	**3.0**
	AF	**3.001**	3.0002	3.54
	VF	0.0	0.0	14.5817

f_{19}	BF	**-3.8628**	**-3.8628**	**-3.8628**
	AF	**-3.8628**	**-3.8628**	**-3.8628**
	VF	0.0	0.0	0.0
f_{20}	BF	**-3.322**	**-3.322**	**-3.322**
	AF	**-3.322**	**-3.322**	**-3.322**
	VF	0.0	0.0	0.0
f_{21}	BF	-10.1532	**-10.1532**	-10.1532
	AF	-10.1416	**-10.1531**	-10.1523
	VF	9.0E-4	0.0	0.0
f_{22}	BF	**-10.4029**	**-10.4029**	**-10.4029**
	AF	-10.4008	**-10.4029**	-10.2963
	VF	0.0	0.0	0.5561
f_{23}	BF	**-10.5364**	**-10.5364**	**-10.5364**
	AF	-10.5309	**-10.5324**	-9.9258
	VF	1.0E-4	4.0E-4	3.5067

Como se puede observar, asignar un número muy pequeño al parámetro $limit$, provoca que los individuos no tengan tiempo suficiente para buscar fuentes de alimento mejores en los alrededores de su fuente de alimento actual (abejas empleadas), o buscar nuevas fuentes de alimento basándose en las mejores fuentes de alimento explotadas actualmente (abejas buscadoras).

Por otro lado, un número excesivamente grande para el parámetro $limit$ puede producir un estancamiento en la búsqueda de nuevas fuentes de alimento, sin que estas sean lo suficientemente buenas para usarlas como referencia.

Tras varios cálculos experimentales, se puede llegar a la conclusión de que el valor de $limit = N * D$ es un buen valor, pues permite la búsqueda de nuevas fuentes de alimento posiblemente mejores cuando la fuente de alimento que explota una abeja no produce mejores resultados significativos, pero dándole tiempo a moverse por el medio para intentar mejorar dicha fuente de alimento.

Debido a que en muchos casos, el cálculo de $limit$ mediante la fórmula $limit = N * D$ produce valores del parámetro cercanos a los valores de los otros dos valores asignados para la comparativa, los resultados en algunos casos pueden ser buenos en ambas columnas, como puede observarse en la tabla.

5.5. Comparativa de Metaheurísticas de dominios continuos

Se escogen para las comparativas poblaciones con 30 individuos, produciendo un número de iteraciones de 1000. Se producen 50 ejecuciones de cada algoritmo para calcular los valores **BF** (mejor *fitness* encontrado en las ejecuciones), **AF** (media de los mejores *fitness* de cada ejecución) y VF (varianza de los valores de los mejores *fitness* encontrados en cada ejecución).

En el algoritmo PSO se elige la topología PSO global, con unos valores de $\alpha = 4$ y $\beta = 1$, tras comprobar buenos resultados con dichos parámetros experimentalmente.

En ABC se escoge como parámetro $limit = N * D$, siendo N el número de individuos de la población y D el número de dimensiones de la función a optimizar.

Por último, el parámetro variable en el algoritmo SSO PF, toma el valor 0.6, que es el que encuentra mejores soluciones dentro de sus posibles valores (comprendidos entre 0 y 1).

Se marcan en negrita los mejores resultados de **BF** y **AF** (en caso de tener en dos o más algoritmos el mismo valor de **BF** o **AF,** se marcarán todos los mejores resultados).

f	Valores	PSO	ABC	SSO
f_1	BF	5.0E-4	**0.0**	9819.4582
	AF	0.2048	**0.0**	10018.6706
	VF	0.0429	0.0	67384.0215
f_2	BF	0.0324	**0.0**	22.3473
	AF	0.4987	0.0	24.943
	VF	1.9988	0.0	10.1331
f_3	BF	**0.0**	**0.0**	**0.0**
	AF	**0.0**	**0.0**	3.0E-4
	VF	0.0	0.0	0.0
f_4	BF	**0.0214**	10.682	69.1463
	AF	**0.1408**	23.0761	72.3469
	VF	0.0083	33.249	2.5904
f_5	BF	28.6642	**0.1495**	2098356.9573
	AF	44.9889	**3.6384**	2372079.7418
	VF	3991.6607	11.5164	9.6499E11
f_6	BF	**0.0**	**0.0**	27848.0
	AF	**0.0**	**0.0**	29298.38
	VF	0.0	0.0	2501415.5873
f_7	BF	**6.0E-4**	0.0717	0.2394
	AF	**0.0044**	0.1389	0.2819
	VF	0.0	0.0016	0.0154
f_8	BF	-10490.208	**-12327.8149**	-2783.6866
	AF	-8301.5404	**-12042.1991**	-2756.0118
	VF	1536175.9265	22440.2087	4054.1175
f_9	BF	2.0985	**0.0**	173.6945
	AF	55.8169	**0.7027**	183.0268
	VF	883.7234	0.5685	77.7493
f_{10}	BF	0.0063	**0.0**	15.563
	AF	0.1789	**1.0E-4**	15.6955
	VF	0.0412	0.0	0.0208
f_{11}	BF	0.0029	**0.0**	75.3571
	AF	0.3615	**0.0056**	82.0991
	VF	0.0783	2.0E-4	65.971
f_{12}	BF	0.0398	**0.0**	6560946.9788
	AF	0.2428	**0.0**	6671926.3937
	VF	0.0695	0.0	1.1311E11

f_{13}	BF	0.574	**0.0**	2.3337E7
	AF	1.6308	**0.0**	2.6291E7
	VF	0.2362	0.0	1.5215E14
f_{14}	BF	**0.998**	0.998	500.0
	AF	**1.8368**	37.287	500.0
	VF	4.3683	13536.0979	0.0
f_{15}	BF	3.0E-4	**4.0E-4**	0.0014
	AF	0.0032	**7.0E-4**	0.0014
	VF	0.0	0.0	0.0
f_{16}	BF	**-1.0316**	**-1.0316**	**-1.0316**
	AF	**-1.0316**	**-1.0316**	**-1.0316**
	VF	0.0	0.0	0.0
f_{17}	BF	**0.3979**	**0.3979**	**0.3979**
	AF	**0.3979**	**0.3979**	**0.3979**
	VF	0.0	0.0	0.0
f_{18}	BF	**3.0**	**3.0**	**3.0**
	AF	**3.0**	3.0002	**3.0**
	VF	0.0	0.0	0.0
f_{19}	BF	**-3.8628**	**-3.8628**	-3.8593
	AF	**-3.8628**	**-3.8628**	-3.8559
	VF	0.0	0.0	0.0
f_{20}	BF	**-3.322**	**-3.322**	-3.1816
	AF	-3.2584	**-3.322**	-3.1532
	VF	0.0039	0.0	0.0021
f_{21}	BF	**-10.1532**	**-10.1532**	-10.1305
	AF	-9.8523	**-10.1531**	-9.9728
	VF	2.2175	0.0	0.0303
f_{22}	BF	**-10.4029**	**-10.4029**	-10.3576
	AF	**-10.4029**	**-10.4029**	-10.2467
	VF	0.0	0.0	0.0768
f_{23}	BF	**-10.5364**	**-10.5364**	-10.4781
	AF	**-10.5364**	-10.5324	-10.3808
	VF	0.0	4.0E-4	0.0913

Se puede comprobar que el algoritmo SSO se queda en la mayor parte de los casos muy por encima de un nivel tolerable de diferencia entre la mejor solución encontrada, y la mejor solución de la función a optimizar, debido a la aleatoriedad de su funcionamiento y la poca capacidad de corrección con solo un parámetro que introduce la probabilidad de repulsión o atracción de las arañas hembra.

Por otro lado, tanto el algoritmo PSO como el algoritmo ABC se quedan cerca de la solución óptima en casi todos los casos, pero cabe destacar que la diferencia más importante está en la varianza de las mejores soluciones encontradas, que indican como de cerca se ha quedado cada ejecución de llegar a la solución óptima. Por tanto, y dependiendo de la función que se esté optimizando, en algunos casos funciona mejor PSO que ABC y viceversa, ya que cada algoritmo se

mueve de forma distinta por el espacio de búsqueda, facilitando en mayor o menor medida el encontrar la solución en un menor número de iteraciones.

5.6. Comparativa de ACO + TSP

El algoritmo ACO aplicado a TSP permite la modificación de diversos parámetros que determinan el comportamiento de las hormigas que configuran la población, y por tanto la facilidad y dificultad de encontrar mejores soluciones en función de dicho comportamiento.

Los grafos utilizados son grafos que representan el TSP clásico: grafos completos [35]. Un grafo completo es aquel cuyos nodos están unidos con todos los demás, es decir, si un grafo posee n nodos, cada uno de los nodos está conectado con los $n-1$ nodos restantes. Por tanto, el número de aristas a de cada grafo es $a = \frac{n*(n-1)}{2}$.

Debido a que los parámetros pueden tomar un rango de valores muy amplio, y que varias combinaciones pueden dar lugar a buenos resultados [8], se prueban varias configuraciones para comprobar el funcionamiento de ACO+TSP.

Se escogen valores de parámetros fijos, como son el número de feromonas iniciales de los caminos del grafo $\tau_0 = 0$ [17], ratio de feromonas dejadas en el camino cuando una hormiga decide pasar por él $q_0 = 0.4$ [17]. Se guarda el mejor camino encontrado hasta el momento, aquel cuyo coste es mínimo. El nodo $destino$ es el siguiente nodo en el camino óptimo que se visita después del nodo $origen$ con una probabilidad de 0.2 [17].

Se escoge una población de 30 individuos para cada ejecución, y cada ejecución produce como máximo 2500 iteraciones. Se muestra el mejor coste encontrado en cada ejecución realizada $\boldsymbol{Coste}$ y el mejor coste encontrado hasta el momento para el problema [11] $\boldsymbol{Mejor\ coste}$.

Se marca en negrita la combinación de parámetro que produce un mejor resultado.

hk48

α	β	ρ	*Coste*	*Mejor coste*
0	1.2	0.7	20513	11461
1	**1.2**	**0.7**	**11470**	
1.5	1.2	0.7	12939	
3	1.2	0.7	15297	

hk48

α	β	ρ	*Coste*	*Mejor coste*
1	1	0.7	12223	11461
1	**1.2**	**0.7**	**11461**	
1	2	0.7	11496	
1	4	0.7	11553	

hk48

α	β	ρ	*Coste*	*Mejor coste*
1	1.2	0	20047	11461
1	**1.2**	**0.2**	**11461**	
1	**1.2**	**0.7**	**11461**	
1	1.2	2	11566	

Se puede observar que son buenos factores $\alpha = 1, \beta = 1.2$ acorde con [8], y que el valor para el factor de evaporación $\rho = 0.7$ produce buenos resultados con $q_0 = 0.4$. Un buen ajuste de los parámetros se conoce con muchas pruebas experimentales, ya que depende de muchos factores que aún con una pequeña variación, pueden producir grandes cambios en el comportamiento del algoritmo, tanto a resultados positivos como negativos.

Tras ellos, se realizan pruebas de varios problemas TSP, recogidos de [11] con estos parámetros $\alpha = 1, \beta = 1.2, \rho = 0.7, q_0 = 0.4$, una buena combinación para los valores de los parámetros según [8], para comprobar como de cerca se quedan de las mejores soluciones encontradas hasta el momento. Cada población está formada por 30 individuos, y se escoge un número máximo de iteraciones de 5000.

Se representa el problema en la columna ***TSP,*** el coste del mejor camino encontrado por el algoritmo implementado en la columna ***CP*** y el coste de la mejor solución conocida [11] en la columna ***CB***.

TSP	***nº nodos***	***nº aristas***	***CP***	***CB***
a280	280	39060	2613	2579
att48	48	1128	33523	10628
att532	532	141246	96347	27686
bays29	29	406	2020	2020
berlin52	52	1326	7916	7542
ch150	150	11175	6668	6528
d657	657	215496	55928	48912
dsj1000	1000	499500	21087873	18659688
f417	417	86736	12424	11861
fl1400	1400	979300	33492	20127
gil262	262	34191	2459	2378
hk48	48	1128	11838	11461
kroA100	100	4950	21285	21282
kroE100	100	4950	22157	22068
lin105	105	5460	14406	14379
p654	654	213531	45375	34643
pr226	226	25425	80703	80369
pr1002	1002	501501	308581	259045
rat99	99	4851	1219	1211

rd100	100	4950	7910	7910
rl1304	1304	849556	275681	252948
si175	175	15225	22761	21407
st70	70	2415	690	675
tsp225	225	25200	3928	3919
u159	159	12561	42640	42080
ulysses16	16	120	73	73
vm1084	1084	586986	259535	239297

6. CONCLUSIONES Y POSIBLES AMPLIACIONES

Se ha logrado el objetivo de crear una web que permita utilizar en un mismo entorno diferentes metaheurísticas basadas en inteligencia de enjambre. A su vez se ha conseguido que la parte gráfica facilite entender cómo evoluciona la ejecución de cada metaheurística, y realizar comparaciones estadísticas entre ellas. En resumen, se ha creado una aplicación interactiva y visual muy útil para cualquier usuario que se encuentre en el proceso de aprendizaje de estas metaheurísticas, así como para sus docentes.

El proyecto demuestra que muchos conocimientos de la carrera son útiles para el futuro pero una conclusión importante que se saca de éste, es que el mundo de la informática está en continuo cambio, que siempre hay que estar aprendiendo, y es esa capacidad de auto-aprendizaje el mayor conocimiento que la titulación ofrece.

Como conclusión final, el proyecto ha resultado una aventura incesante de búsqueda de información y tecnologías adecuadas para el desarrollo del mismo. Prácticamente todas las herramientas utilizadas han pasado por un proceso de aprendizaje para los autores, así como un proceso de adaptación al proyecto, y el hecho de que se haya conseguido un producto agradable a la vista, funcional e intuitivo es una gran satisfacción.

Al tratarse de una aplicación con un fin altamente educativo, pueden realizarse ampliaciones en función de numerosos factores. A continuación se lista una serie de posibles ampliaciones:

- Actualmente, existen múltiples algoritmos de optimización, además de los añadidos a la aplicación. Una posible ampliación del proyecto incluye la adición de nuevos métodos de optimización. Uno de los posibles métodos de optimización a añadir sería incluir algoritmos genéticos [37, 38].

- Además de las funciones de minimización predeterminadas añadidas al proyecto, existe la posibilidad de añadir nuevas funciones de minimización por defecto.

- Puede añadirse la posibilidad de no solo ejecutar la optimización de algoritmos de minimización al proyecto, si no también añadir funciones cuya optimización se produzca mediante la maximización de la función, debiendo añadir, por tanto, la posibilidad de escoger tanto la minimización, como la maximización del algoritmo elegido.

- Posibilidad de modificar los distintos parámetros internos de los algoritmos implementados. Ej.: Posibilidad de elegir entre las múltiples formas de variación del factor inercial en el algoritmo PSO.

- La aplicación permite actualmente representación interactiva de los resultados en dos dimensiones, pero una posible mejora es la de añadir dicha representación para funciones de 3 dimensiones, con una cámara que permita movernos a lo largo de los ejes X, Y, Z

BIBLIOGRAFIA

[1] Hu, X. (2006). PSO tutorial. URL: http://www. swarmintelligence. org-/tutorials. php.

[2] Karaboga, D., Aslantas, V., Karaboga, N., Basturk, B., Okdem, S., Ozturk, C., Koylu F., Gorkemli, B. y Hancer, E. Software. Artificial Bee Colony (ABC) Algorithm. URL: http://mf.erciyes.edu.tr/abc/

[3] Cuevas, E., Cienfuegos, M., Zaldívar, D., & Pérez-Cisneros, M. (2013). A swarm optimization algorithm inspired in the behavior of the social-spider. Expert Systems with Applications, 40(16), 6374-6384.

[4] Karaboga, D. (2010). Artificial bee colony algorithm. scholarpedia, 5(3), 6915.

[5] Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary programming made faster. Evolutionary Computation, IEEE Transactions on, 3(2), 82-102.

[6] Bansal, J. C., Singh, P. K., Saraswat, M., Verma, A., Jadon, S. S., & Abraham, A. (2011, October). Inertia weight strategies in particle swarm optimization. In Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on (pp. 633-640). IEEE.

[7] Nieto, J. G., & Torres, E. A. (2006). Algoritmos basados en cúmulos de partículas para la resolución de problemas complejos. Septiembre de 2006. URL: http://neo.lcc.uma.es/staff/jmgn/doc/Memoria_PFC_JMGN.pdf

[8] Erol, A. H., Er, M., & Bulkan, S. (2012). Optimizing the Ant Colony Optimization Algorithm Using Neural Network for the Traveling Salesman Problem. Actas de la Conferencia Internacional de 2012 sobre Ingeniería Industrial y Gestión de Operaciones. Estambul, Turquía, 3-6 Julio de 2012.

[9] Dorigo, M. (2007). Ant colony optimization. Scholarpedia, 2(3):1461

[10] Bacon, D. Expr. Parser Package. URL: https://github.com/darius/expr

[11] TSP. URL: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

[12] Shafranovich, Y. (2005). Common format and MIME type for Comma-Separated Values (CSV) files. URL: http://tools.ietf.org/html/rfc4180.html

[13] Kiran, M. S., & Gündüz, M. (2014). The Analysis of Peculiar Control Parameters of Artificial Bee Colony Algorithm on the Numerical Optimization Problems. Journal of Computer and Communications, 2(04), 127.

[14] Kennedy, J. (2010). Particle swarm optimization. In Encyclopedia of Machine Learning (pp. 760-766). Springer US.

[15] Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of global optimization, 39(3), 459-471.

[16] Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: artificial bee colony (ABC) algorithm and applications.Artificial Intelligence Review, 42(1), 21-57.

[17] Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 26(1), 29-41.

[18] Dorigo, M., & Birattari, M. (2010). Ant colony optimization. In Encyclopedia of Machine Learning (pp. 36-39). Springer US.

[19] cnpso Particle Swarm Otimization Algorithm. URL: https://code.google.com/p/cnpso/

[20] Karaboga, D., Aslantas, V., Karaboga, N., Basturk, B., Okdem, S., Ozturk, C., Koylu F., Gorkemli, B. y Hancer, E. Software. URL: http://mf.erciyes.edu.tr/abc/software.htm

[21] Borgelt C. (2005). ACOpt - Ant Colony Optimization Demonstration. URL: http://www.borgelt.net/acopt.html

[22] Gosling, J. (Ed.). (2000). The Java language specification. Addison-Wesley Professional. URL: http://docs.oracle.com/javase/specs/jls/se8/jls8.pdf

[23] Álvarez, M. (2002). La tecnología Java para la creación de páginas web con programación en el servidor. URL: http://www.desarrolloweb.com/articulos/831.php

[24] Java Servlet Tecnology. URL: http://www.oracle.com/technetwork/java/index-jsp-135475.html

[25] Interface Servlet. URL: http://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/Servlet.html

[26] Mora, S. L. (2001). Programación en Internet: clientes web. Editorial Club Universitario. URL: http://rua.ua.es/dspace/bitstream/10045/16994/1/sergio_lujan-programacion_en_internet_clientes_web.pdf

[27] JavaScript. URL: https://developer.mozilla.org/es/docs/Web/JavaScript

[28] Eguíluz Pérez, J. (2009). Introducción a JavaScript. URL: http://librosweb.es/javascript/index.html

[29] Eguíluz Pérez, J. (2008). Introducción a CSS. España. URL: http://librosweb.es/css/

[30] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (1998). Extensible markup language (XML). World Wide Web Consortium Recommendation REC-xml-19980210. URL: http://www.w3.org/TR/1998/REC-xml-19980210

[31] Crockford, D. (2006). The application/json media type for javascript object notation (json). URL: http://tools.ietf.org/html/rfc4627

[32] Otto, M. Thornton, J. Bootstrap 3, manual oficial. URL: http://librosweb.es/bootstrap_3/

[33] Chaffer, J. (2009). Learning JQuery 1.3: Better Interaction and Web Development with Simple JavaScript Techniques. Packt Publishing Ltd. URL: http://books.google.es/books?hl=es&lr=&id=R9JwBcAUHaoC&oi=fnd&pg=PT6&dq=jQuery&ots=rbyOLMyCJM&sig=ehIKd1zMlc0tXooKC3x4VjpQB9s#v=onepage&q=jQuery&f=false

[34] Facultad de Ingeniería. Universidad Nacional de San Juan. Unidad temática XIII. Funciones de una sola variable. URL: http://www.fi.unsj.edu.ar/asignaturas/ingprocesosAlimentos/APUNTES%20PDF/UNIDAD%20XII.pdf

[35] Souza, A. (Winter Term 2010-2011). Combinational algorithms. Chapter 11. Traveling Salesman. (pp. 75-77). Humboldt University Berlin.

[36] Cantelon, M., Holowaychuk, T. J., Rajlich, N., & Harter, M. (2014). Node. js in Action. Manning Publications. URL: http://toc.dreamtechpress.com/toc_978-93-5119-260-2.pdf

[37] Miller, J. F., & Thomson, P. (2000). Cartesian genetic programming. In Genetic Programming (pp. 121-132). Springer Berlin Heidelberg.

[38] Banzhaf, W., Nordin, P., Keller, R. E., & Francone, F. D. (1998). Genetic programming: an introduction (Vol. 1). San Francisco: Morgan Kaufmann.

[39] Marcelionis, A.amcharts. (2004). URL: http://www.amcharts.com/about/

[40] Preston-Werner, T., Wanstrath, C., Hyett, PJ. Github. URL: https://github.com/about

[41] Apache™ Subversion®. URL: http://subversion.apache.org/

[42] Apache Tomcat. URL: http://tomcat.apache.org/

[43] D'Anjou, J. (Ed.). (2005). The Java developer's guide to Eclipse. Addison-Wesley Professional. URL: http://books.google.es/books?hl=es&lr=&id=6Ob1ANNVcXcC&oi=fnd&pg=PR5&dq=eclipse+java&ots=jnBq4PQlzy&sig=Ebnk4yvr-Lskf0Dd3kgkiQPLWhM#v=onepage&q=eclipse%20java&f=false

[44] FireBug. URL: http://getfirebug.com/whatisfirebug

[45] PhotoShop CS5. URL: http://www.adobe.com/products/photoshop.html?promoid=GWQSR

[46] Ho, D. NotePad++. URL: http://notepad-plus-plus.org/

[47] TeamViewer. URL: http://www.teamviewer.com/es/Index.aspx